PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 29/06

A1

(11) International Publication Number: WO 98/28891

(43) International Publication Date: 2 July 1998 (02.07.98)

(21) International Application Number: PCT/US97/22688

(22) International Filing Date: 16 December 1997 (16.12.97)

(30) Priority Data:
08/772,712 23 December 1996 (23.12.96) US

(71) Applicant: SCHLUMBERGER TECHNOLOGY CORPORATION [US/US]; 8311 North R.R. 620, P.O. Box 200015, Austin, TX 78720-0015 (US).

(72) Inventors: ANIGBOGU, Julian, C.; 12013 Grey Fawn Path, Austin, TX 78750-1428 (US). RENISKA, Kim; 7501 Lakewood Drive, Austin, TX 78750 (US).

(74) Agent: MASELES, Danita, J., M.; Schlumberger Austin Product Center, 8311 North R.R. 620, P.O. Box 200015, Austin, TX 78720-0015 (US).

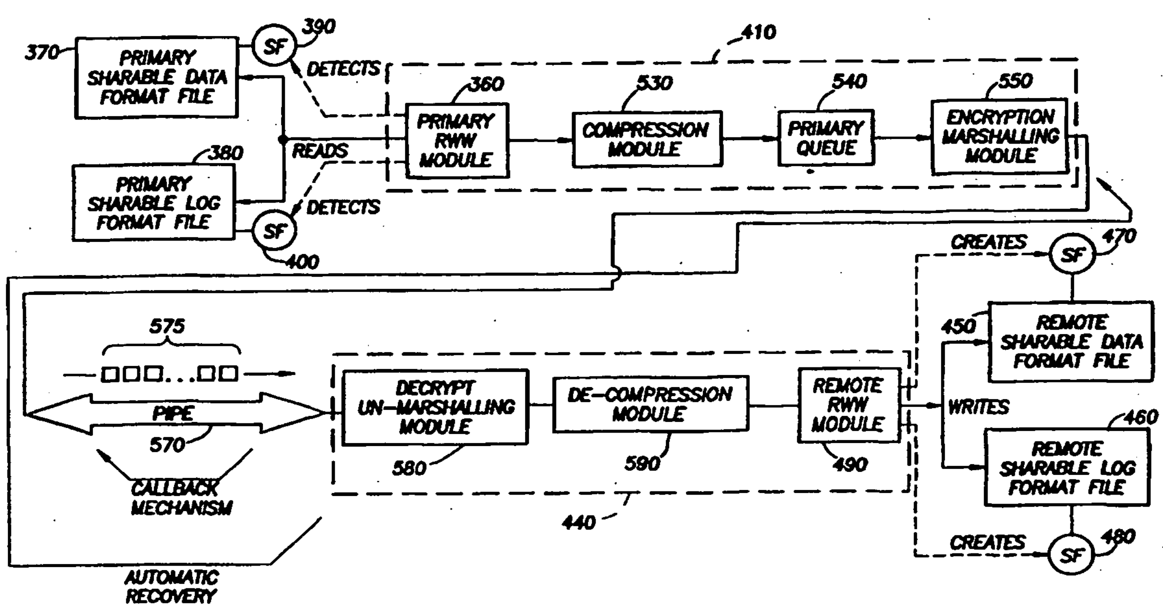
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published

With international search report.
With amended claims and statement.

Date of publication of the amended claims and statement: 3 September 1998 (03.09.98)

(54) Title: AN APPARATUS, SYSTEM AND METHOD FOR SECURE, RECOVERABLE, ADAPTABLY COMPRESSED FILE TRANSFER



(57) Abstract

The present invention provides a system for handling and transmitting a file over a communication channel wherein the file may be adaptably compressed to improve throughput. The invention also provides for a method of recovery in the event of a communication failure. The file may be encrypted while it is being transmitted. The compression and transmission may occur while the file is being written, so that the receiving location receives the data in near real time.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

An Apparatus, System and Method for Secure, Recoverable, Adaptably Compressed File Transfer

TECHNICAL FIELD

This invention relates in general to the field of file compression and transfer, and more particularly to an improved apparatus, system and method of achieving secure, recoverable, adaptably compressed file transfers.

BACKGROUND OF THE INVENTION

The ability to move data reliably and securely from one location to another has become an important key to success, and in many cases to survival, for many companies and businesses. In businesses for which the primary product or service delivered is data, this ability is even more critical.

Recent years have seen a proliferation of computers and people connected on a network, in one form or another. Companies have come to depend on the ability to move data on the network to accomplish their tasks and to ensure the continuation of their businesses. A wide variety of connections and speeds are inherent on these networks, ranging from cellular phone modems and dial-up low-speed connections up through ISDN lines, T1 dedicated lines, and high speed ATM connections. A wide variety of different computers with different processing speeds and communications capabilities are attached as nodes on these networks. In many cases, especially in the oil and gas industry, many of the computers are mobile and come and go (attach and detach) from the networks from different locations.

The need to transmit information on these widely varying networks has created a demand for methods, processes, and standard techniques for moving data from one computer system to another in a secure, efficient, and reliable manner. Many of the lower to mid-level protocols have been accepted as standards. Among the most notable of these is the Transport Control Protocol/Internet Protocol (TCP/IP). This is the basic method for moving packets of data on standard networks, including the Internet.

While there have been various applications developed using TCP/IP for transmission, the most widely known is the File Transfer Protocol ("FTP"). FTP was developed to support many different hardware and operating system platforms and is widely used to move data files around networks. It works fairly well in well established, reliable networks but has some limitations on noisy, unreliable, very low bandwidth network connections. Another disadvantage of FTP is that FTP does no compression of data, so all data must be transmitted as is. FTP also provides no recovery mechanism

for file transfers. This means that if the connection is lost during a file transmission, the file transfer must be restarted from the beginning. FTP has no inherent security mechanism for protecting the data "on the wire" during the transmission. FTP also requires that the all data to be transmitted is available, *i.e.* that the complete file is available, before transmission can begin. Yet another disadvantage of FTP is that FTP requires that the file is not being written when the transmission begins or is in progress.

One of the many needs for secure, recoverable, adaptively compressed file transfers may be found in the oil and gas industry. In the oil and gas industry, operating companies which own and/or manage hydrocarbon wells evaluate the wells by wireline logging. In wireline well logging, one or more tools are connected to a power and data transmission cable or "wireline" and are lowered into the well borehole to obtain measurements of geophysical properties for the area surrounding the borehole. The wireline supports the tools as they are lowered into the borehole, supplies power to the tools and provides a communication medium to send signals to the tools and receive data from the tools. Commonly, tools are lowered to a depth of interest in the well and are then retrieved. As the tools are retrieved, they send data about the geological formations through which they pass through the wireline to data acquisition and processing equipment at the surface, usually contained inside a logging truck or a logging unit.

The data acquisition and processing equipment, including software, compiles the data from the tools into a "log," a plot which presents the geophysical information concerning the geological formations encountered by the well, frequently by depth. Logs can also be used to evaluate current production from producing wells or to inspect the integrity of production equipment in a producing well. In any case, the data gathered during the logging operation is generally presented on the log by depth, but may also be presented by time, or any other index by which multiple physical entries are recorded. U.S. Patent No. 5,051,962 (incorporated by reference) describes such a well logging system controlled by a general purpose computer programmed for real time operation. Various data acquisition and processing software programs are known in the art. An example of data acquisition and processing software is Schlumberger's proprietary MAXIS™ system, which is a suite of separate computer programs.

The data acquisition and processing software writes the log data to two types of locked format files on disk. By "locked," it is meant that the format files cannot be written to and read from at the same time. The two types of locked format files are distinguished by the type of information they contain: one is a data format file and the other is a graphics format file. The data format file contains the numerical properties of the log data; the graphics format file contains the pictorial representation of the data.

The data acquisition and processing software continues writing the log data to the locked data format file and the locked graphics format file until the log is complete. Then the data from the locked data format file and the locked graphics format file may be translated from digital readings into physical form by a marking device such as a printer. In addition to the locked data format file and the locked graphics format file, the data acquisition and processing software may send the log data to a viewing monitor, via a renderer. Using the monitor, the well logging professional ("logging engineer") conducting the logging operation can view the log as it is being compiled.

After the log is compiled, it may be transmitted to the operating company's headquarters for interpretation and review by management. The paper log may be sent directly from the wellsite to the operating company as a facsimile. Alternatively, the completed locked data format file may be sent from the wellsite to a data processing center via satellite using a protocol such as DECNET. The data processing center could in turn transmit the log as a facsimile to the operating company. As another alternative, the completed locked data format file may be sent from the wellsite to an operating company using a computer program such as Blast™ by U.S. Robotics.

The data acquired by logging is often crucial to the decision-making process on what will be done with the well being logged. Take, for example, a well which has just been drilled and logged. Depending on the results of the log, the well could be drilled deeper, plugged and abandoned as non-productive or cased and tested -- or perhaps the decision will be that additional logs are required before the decision on the disposition of the well can be made. The results of the log may also help determine whether the well requires stimulation or special completion techniques, such as gas lift or sand control. In any case, these decisions are critical and have to be made very quickly. Mistakes or even mere delay can be extremely expensive.

Because log interpretation is part art and part science, the operating company which is drilling or producing the well frequently desires to have its own personnel viewing the log data as the well is being logged. But the operating company may be located half a world away from the well itself. Drilling and production activities are often located in remote locations and it is difficult for the operating company to have its own personnel, such as a geologist or petrophycist, join the wireline company's logging engineer on site during the logging operation. Sometimes logistics or severe weather conditions prevent the operating company from sending anyone to the wellsite for the logging operation. Furthermore, sending personnel to wellsites is expensive and exposes them to all of the hazards of the drilling or production operation, as well as the hazards and inconvenience of travel. As a consequence, tentative decisions often have

to be made before the operating company can complete its own review of the actual logging data, relying solely on the interpretations conducted at the wellsite.

Accordingly, a need exists for a system or method which would allow files to be transferred securely, especially while "on the line."

A further need exists for a system or method which would allow a file to be transferred while making maximum use of low bandwidth connections.

A further need exists for a system or method which would allow a file to be transferred while adaptably compressing the file to improve transmission throughput.

A further need exists for a system or method which would overcome the disadvantages of the File Transfer Protocol.

A further need exists for a system or method which would allow files to be transferred taking into account the unique requirements of mobile network connections.

A further a need exists for a system or method which would allow files to be transferred as they are compiled in at least near real time from one location to a remote location remote from the primary for viewing or other use.

A further need exists for a system or method which would allow well data files to be transferred as they are compiled in at least near real time from a wellsite to a remote location remote from the well site for viewing or other use.

A further need exists for a system for or method of file transfer which would provide a recovery method should communications be lost.

Because the data from the logging operation is of a highly competitive nature and is extremely confidential, a need exists for a system or method which will send well data files from a wellsite to a remote location in near real time, in such a way that the data files are not susceptible to being misdirected or lost.

A further need exists for a system or method which can maintain the confidentiality of the well data while it is being transmitted.

A further need exists for a system or method of transferring files in near real time from one location to a remote location so that so that persons can view the files in near real time, without the expense of travelling to the primary location.

A further need exists for a system or method of transferring well data files in near real time from wellsite to a remote location remote from the wellsite so that persons can view the well data files near real time as they are being compiled, without the expense of travelling to the wellsite and without being exposed to the hazards of the wellsite.

SUMMARY OF THE INVENTION

In accordance with the present invention, an apparatus, system and method are provided that substantially eliminate or reduce the disadvantages and problems associated with the previously developed file transfer systems.

The present invention provides a system for handling and transmitting a file over a communication channel which includes a file having a degree of compressibility; a communications channel having a physical bandwidth; a file transfer server at a first location; a file transfer client at a second location and a means for compressing the file based on the physical bandwidth, the capabilities of the transmitting and receiving processors and the degree of compressibility of the file.

The present invention also provides for a system for managing and transmitting a file over a communication channel which includes a file having a degree of compressibility and a plurality of buffers, a communications channel having a physical bandwidth;, a file transfer server at a first location, a file transfer client at a second location, a feedback loop for optimally compressing the file for transmission, including a means for compressing a first buffer to a first level of compressibility; a means for evaluating the efficiency of the compression of the first buffer, a means for adjusting the compression of second buffer based on the evaluation of the compression of the first buffer.

The present invention also provides for a system for managing and transmitting a file over a communication channel which includes a file and a means for compressing buffers of the file in a stream in real time while it is being written.

The present invention also provides for a system for managing and transmitting a file over a communication channel which includes a file, a transmission of the file having a state and a location, and a means for maintaining the state and location of the transmission within the file so that transmission can be resumed in the event of an interruption at the point of the interruption. The interrupted transmission may be automatically resumed or manually resumed.

The present invention also provides for a system for managing and transmitting a file over a communication channel which includes a file, a means for encrypting the file in a stream before and during transmission, and means for de-crypting the file after receipt of transmission.

The present invention also provides for a system for managing and transmitting a file over a communication channel which includes a file having a degree of compressibility and a plurality of buffers, a communications channel having a physical bandwidth, a transmitting processor, a receiving processor, a means for optimally

compressing the file based on the physical bandwidth, the capabilities of the transmitting and receiving processors and the degree of compressibility of the file, a means for compressing the buffers of the file in a stream in real time while the file is being written, a transmission of the file having a state and a location, a means for maintaining the state and location of the transmission within the file so that transmission can be resumed in the event of an interruption at the point of the interruption, a means for encrypting the file in a stream before and during transmission; and a means for de-crypting the file after receipt of transmission.

The present invention also provides for a method of compressing a source file, having a plurality of buffers, for transmission over a communication channel, which includes selecting a first buffer of the source file, compressing the first buffer to a first compression level, marshalling the first buffer, transmitting the first buffer, de-compressing the first buffer, writing the first buffer to a destination file, determine a first throughput which was used for steps (a) through (f), selecting the compression level of a second buffer based on the first throughput, repeating steps (a) through (h) for each of the buffers in turn until all of the buffers of the source file have been transmittedand have been written to the destination file. The marshalling steps may include encrypting the buffers and the de-compressing steps include de-encrypting the buffers. There may also be a further step of writing to the source file while performing one or more of steps (a) through (i).

The present invention also provides for a system for handling and transmitting a first file over a communication channel which includes a means for reading while writing a first file, a means of compressing the first file, a means of marshalling the first file, a means for transmitting the first file from a first location to a second location, a means for unmarshalling the first file, a means for decompressing the first file, a means for writing the first file to a second file. The means a means for reading while writing a first file, a means of compressing the first file and means of marshalling the first file may comprise a file transfer server. The means for unmarshalling the first file, a means for decompressing the first file, and a means for writing the first file to a second file may be a file transfer client. The file transfer server may include a read while write module and/or a compression module. The compression module may use a deflation algorithm and or a Huffman tree.The compression module may compress in a first stage to produce literals and pointersand may compress in a second stage to produce a Huffman tree and a block. The invention may also include a means for encrypting the first file before marshalling the first file and a means for decrypting the first file after the first file is transmitted. The file transfer client may include a read while write module. The first file may be a sharable file.

The present invention also provides for a system for handling and transmitting a first file over a communication channel which includes a means for reading while writing data to the first file, a means for compressing the first file, a means for queueing the first file, a means for marshalling the first file, a means for transmitting the first file from a first location to a second location, a first means for unmarshalling the first file, a means for re-queueing the first file, a means for re-marshalling the first file, a remote procedure call for sending the first file within the second location, a second means for unmarshalling the first file, a means for decompressing the first file, a means for writing the first file to a second file. The means for reading while writing data to the first file, for compressing the first file, for queueing the first file, and for marshalling the first filemay be a first file transfer server. The means for a first means for unmarshalling the first file, a means for re-queueing the first file, a means for re-marshalling the first filemay be a file transfer client. The file transfer server may include a read while write module. The file transfer server may include a compression module.The compression module may use a deflation algorithm or aa Huffman tree. The compression module may compress in a first stage to produce literals and pointers. The compression module may compress in a second stage to produce a Huffman tree and a block. The present invention may further include a first means for encrypting the first file before marshalling the first file and a first means for decrypting the first file after the first file is transmitted. The present invention may further include a second means for encrypting the first file after the first file is transmitted and a a first means for decrypting the first file after the first file has been sent through the remote procedure call within the second location. The means for a second means for unmarshalling the first file, a means for decompressing the first file, a means for writing the first file to a second file may be a second file transfer server. The file transfer client may include a read while write module. The first file is a sharable file. The second file may be a sharable file.

The present invention also provides for a method for handling and transmitting a first file over a communication channel including the steps of reading while writing data to the first file, compressing the first file, queueing the first file, marshalling the first file, transmitting the first file from a first location to a second location, unmarshalling the first file, decompressing the first file, and writing the first file to a second file. The steps of reading while writing data to the first file, compressing the first file, queueing the first file, marshalling the first file may be accomplished by a file transfer server. The steps of unmarshalling the first file, decompressing the first file, and writing the first file to a second file may be accomplished by a file transfer client. The file transfer server may include a read while write module. The file transfer server may include a compression module. The compression step may include the use of a deflation

algorithm for a first stage of compression or a Huffman tree for a second stage of compression. The use of the deflation algorithm in the first stage may produce literals and pointers. The use in a second stage of the Huffman tree may produce a Huffman tree and a block. The present invention may also include the futher include the steps of encrypting the first file before marshalling the first file and decrypting the first file after the first file is transmitted. The file transfer client may include a read while write module. The first file may be a sharable file. The second file may be a sharable file.

The present invention also provides for a method for handling and transmitting a first file over a communication channel including the steps of reading while writing data to the first file, compressing the first file, queueing the first file, marshalling the first file, transmitting the first file from a first location to a second location, unmarshalling the first file, re-queueing the first file, re-marshalling the first file, sending via a remote procedure call the first file within the second location, unmarshalling the first file a second time, decompressing the first file, and writing the first file to a second file. The steps of reading while writing data to the first file, compressing the first file, queueing the first file, marshalling the first file may be accomplished by a first file transfer server. The steps unmarshalling the first file, re-queueing the first file, re-marshalling the first file may be accomplished by a file transfer client. The file transfer server may include a read while write module.The file transfer server includes a compression module.The method for handling and transmitting a first file as in claim 52 wherein the compression step includes use of a deflation algorithm for a first stage of compression. The compression step uses a Huffman tree for a second stage of compression. The use of the Huffman tree produces a Huffman tree and a block.T he use of the deflation algorithm in the first stage produces literals and pointers.The present invenrtion may also futher comprise the steps of encrypting the first file before marshalling the first file, decrypting the first file after the first file is transmitted to the second location.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a primary location in communication with a remote location according to the present invention.

FIG. **2** illustrates the data acquisition and processing equipment of the primary location, including inputs and outputs.

FIG. **3** illustrates the remote location equipment, including inputs and outputs.

FIG. **4A** illustrates the components of the primary memory.

FIG. **4B** illustrates the components of the remote memory.

FIG. **5A** illustrates the data acquisition and processing software and other software programs at the primary location and the file transfer to the remote location. FIG. **5B** illustrates the the remote data processing software and other software programs at the remote location and the file transfer to the primaty location.

FIG. **6** illustrates the transmission process if near real time display is not desired.

FIG. **7A** and FIG. **7B** illustrate the transmission process if near real time display is desired.

FIG. **8** illustrates the communication system.

FIG. **9** illustrates the compression details.

FIG. **10** illustrates adaptable compression.

FIG. **11A - 11F** illustrate the different scenarios for file transfer.

DESCRIPTION OF A PREFERRED EMBODIMENT

The above-noted and other aspects of the present invention will become more apparent from a description of a preferred embodiment, when read in conjunction with the accompanying drawings. The drawings illustrate the preferred embodiment of the invention. In the drawings, the same members have the same reference numerals.

I. Overview

As described in co-pending U.S. Patent Application Serial Number ___ and as illustrated in FIG. 1, during logging operations, log data is sent from a logging tool **10** through wireline **20** to a data acquisition and processing system **30** at a wellsite or primary location **40**. ("Primary is used in the sense of the word "first," not "most important.") A primary input device **50**, such as a keyboard, allows human input into the data acquisition and processing system **30**. Outputs to the data acquisition and processing system **30** include a primary monitor **60**, and a primary printer **70**, which acts through a primary operating system device driver **80** to print a log **90**. The data acquisition and processing system **30** communicates with a remote location **100** through a communication channel **110**. The remote location **100** has a remote location equipment **120**, a remote input device **130** for human input, such as a keyboard, and outputs, such as a remote monitor **140** and a remote printer **150** which acts through a remote operation system device driver **160** to produce a log identical to the log **90** produced at the primary location. (Although the logs have separate physical existences, both logs are given the reference number **90** to indicate this identity.)

Continuing to refer to FIG. **1**, the data acquisition and processing equipment **30** and the remote location equipment **120** establish communication over the communication system **110**. Preferably, the communication system **110** uses TCP/IP protocol and is based on a physical link, such as hard-wire, cellular, radio, microwave, satellite or telephone transmission. In alternative embodiments, other types of point-to-point network protocols, such as Blast by U.S. Robotics, may be used. In still other embodiments, other types of communication systems could be used. If using the TCP/IP protocol, both the data acquisition and processing equipment **30** and remote location equipment **120** have individual IP addresses **102, 104** in accordance with the TCP/IP protocol. The bandwidth of the communication system **110** is preferably at least 10 kilobits per second. As illustrated in FIG. 8, the components of the communication system **110** include a physical link **111**, the protocol **112** such as TCP/IP, and the transport mechanism **113**, as well as other components illustrated.

FIG. **2** illustrates the data acquisition and processing equipment present at the primary location **40**, including its inputs and outputs. As illustrated in FIG. **2**, the log

data enters the data acquisition and processing equipment **30** through a data acquisition system **170.** The data acquisition and processing equipment **30** also includes a primary data processor **180**, a primary bus **190**, a primary file system **195**, and a primary memory **200.** The primary input device **50**, the primary monitor **60**, the primary printer **70**, the primary operating system device driver **80**, and the log **90** are also illustrated.

FIG. **3** illustrates the remote location equipment **120**, including its inputs and outputs. As illustrated in FIG. **3**, the remote location equipment **120** includes a remote data processor **210**, a remote bus **220**, a remote file system **225**, and a remote memory **230.** The remote input device **130**, the remote monitor **140**, the remote printer **150**, the remote operating system device driver **160**, and the log **90** are also illustrated. The remote data processor **210** is preferably a Pentium PC (P5-90 or higher), with an Ethernet interface or a modem. The remote memory **230** has preferably at least 32 Megabyte RAM.

FIG. **4A** illustrates the components of the primary memory **200**. As illustrated in FIG. **4A**, the primary memory **200** includes a primary data acquisition and processing software **240**, a primary operating system software **250**, a primary data storage **260**, a primary file transfer utility program **270** and other primary software **280**. The primary operating system software **250** is preferably Windows® NT™ by Microsoft Corporation. If near real time viewing is not desired, Open VMS or other operating systems may be used.

FIG. **4B** illustrates the components of the remote memory. As illustrated in FIG. **4B**, the remote memory **230** includes a remote data processing software **290**, a remote operating system software **300**, a remote data storage **310**, a remote file transfer utility program **320** and other remote software **330**. The remote operating system software **300** is preferably Windows® NT™ by Microsoft Corporation.

FIG. **5A** illustrates the data acquisition and processing software **240** and other software programs at the primary location and FIG. **5B** illustrates the remote data processing software **290** and other software programs at the remote location. As illustrated by FIG. **5A**, as log data enters the data acquisition and processing software **240**, the data enters a primary data formatting program **340** where it is formatted as numerical data in industry standard format for storage. The data also enters a log generating program **350** which adds commands and other instructions to the data to create graphics data. The numerical data includes numerical properties of the data; the graphics data includes pictorial representation of the data.

The data formatting program **340** and the log generating program **350** act through a primary read-while-write ("RWW") module **360** to write the data as it is

received to a primary sharable data format file **370** and to one or more primary sharable log graphics files **380**, respectively. (Different primary sharable log graphics files **380** may be created for different presentations of the same log data.)

Unlike a locked format file, a sharable format file allows shared file access, such that the sharable format file may be written to and read from at the same time. Preferably, each sharable format file contains all of the acquired data and input parameters for a single logging run, that is, for the data from a set of logging tools run into the well simultaneously.

As the primary RWW module **360** begins to write logging data to the primary sharable data format file **370** and the primary sharable graphics format file **380**, the primary RWW module **360** also creates for each of the primary sharable data format file **370** and for the primary sharable graphics format file, **380**, a primary semaphore file, **390, 400** respectively. The primary semaphore files **390**, **400** have the same name as the primary sharable format file for which they were created, with the addition of a "_smf" at the end. The existence of a semaphore file indicates that the primary sharable format file with the similar name is a sharable file. Any program which shares access to the sharable format file also uses a copy of the primary RWW module to check for the existence of the associated semaphore file to determine whether the format file is sharable.

Refering to FIG. **5A**, the primary file transfer utility program **270** may include a primary file transfer server **410** (also illustrated in FIG. **5B**) and a primary file transfer client **420**. Refering to FIG. **5B**, the remote file transfer utility program **320** may include a remote file transfer server **430** and/or a remote file transfer client **440**. The minimum needed is a file transfer server in either the primary or remote location and a file transfer client in the other location. But there may be a file transfer server and a file transfer client in each location.

The primary file transfer server and the remote file transfer server are always on and are always listening for requests from one of the file transfer clients. The file transfer clients are not always on, but are send requests or queries to the file transfer server as initiated by a request from the user through the input device. One file transfer server may handle requests from more than one file transfer client. File transfer clients may or may not read from or write to one of the format files. A file transfer server reads from and writes to a format file. Different scenarios for data requests between file transfer clients and servers are found in FIG. **11**.

II. The Read Process

After establishing communication, the remote file transfer client **440** requests data, either numerical data, graphics data or both, from the primary file transfer server **410**. Depending on which data it is seeking, the primary file transfer server **410**, which contains a copy of the primary RWW module **360**, checks for primary semaphore files **390**, **400** for both or either of the primary sharable data format file **370** and the primary sharable graphics format file **380**. (For example, the primary file transfer server **410**, using the primary RWW module **360**, would check for primary semaphore file **390** for the primary sharable data format file **370**.) After verifying that the primary sharable data format file **370** and/or the primary sharable graphics format file **380** are sharable, the primary file transfer server **410**, using its copy of the primary RWW module **360**, begins to read the data written the primary sharable data format file **370** and/or the primary sharable graphics format file **380**. The primary file transfer server **410** starts at the beginning of the data in the primary sharable data format file **370** and/or the primary sharable graphics format file **380** and continues to read a certain amount of data (as described below) at a time until it reaches a last datum written. Then the primary file transfer server **410** continues to read additional data as it is being written to the primary sharable data format file **370** and/or the primary sharable graphics format file **380** until the logging operation is completed. At that time, the primary sharable data format file **370** and the primary sharable graphics format file **380** are closed by the data acquisition and processing software **240**. The data acquisition and processing software **240** also closes and deletes the similarly-named semaphore files **390**, **400**.

III. Compression and Decompression

The primary file transfer server **410** reads approximately 32,000 bytes (or 32 kilobytes) of data at a time into a buffer, which is then compressed. The compression and decompression functions are performed by a "compression module" and a "decompression module." The compression module and the decompression module use variations of the deflation/inflation algorithms as described in Ziv J., Lempel A., "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343 (1977) ("LZ77"), incorporated herein by reference.

As described below, compression is accomplished on two stages.

A. Detailed Description of Compression: First Stage

1. <u>The Deflation Algorithm</u>

a. Overview

As illustrated in FIG. **9**, the compression module uses a deflation algorithm **450** which is a variation of that described in LZ77. The deflation algorithm **450** finds duplicated strings of data in the input data using a fixed-length sliding window. In this description, "string" must be taken as an arbitrary sequence of bytes, and is not restricted to printable (ASCII) characters. Preferably, the string is at least three bytes long. As the fixed length sliding window slides along the data, it looks for re-occurrences of a string of data. The re-occurrence of the string is replaced by a pointer to the previous string. The pointer is expressed as a pair of numbers representing distance and length. Distance means the distance from the beginning of one string and the beginning of its duplicate string. Length is the length of the string. Preferably, distances are limited to 32K bytes, while lengths are limited to 258 bytes. When a string does not occur anywhere in the previous 32K bytes, the string is emitted as a sequence of literal bytes. A sequence of literal bytes is not compressible.

In still greater detail, the deflation algorithm actually finds the duplicate strings using a hash table. As the fixed length sliding window slides over the data, all input strings of at least three bytes in length are inserted into the hash table. An entry into the hash table is called a hash chain. The deflation algorithm computes a hash index, which points to the location of the hash chain, for the next three bytes. If the hash chain for this index is not empty, all strings in the chain are compared with the current input string, and the longest match is selected.

The hash chains are searched starting with the most recent strings, to favor small distances and to take advantage of Huffman encoding, described below. The hash chains are singly linked. There are no deletions from the hash chains; the deflation algorithm simply discards matches that are too old.

To avoid a worst-case situation, very long hash chains are arbitrarily truncated at a certain length, determined by a runtime option. That is, one may select a level of compression, preferably Levels 1 to 9, to determine at what length to truncate the hash chain. Each level will discard hash chains of specific lengths. Because the very long hash chains may be truncated, the compression module does not always find the longest possible match. The compression module generally finds, however, a match which is long enough. Level 0 implements another algorithm (a "stored algorithm") which consists of only writing header information into the output buffer and then copying input to output unchanged, i.e. there is no compression.

b. Lazy Evaluation Method

The compression module may also defer the selection of the matches with a lazy evaluation mechanism. The lazy evaluation mechanism is preferably used for Levels 4

and higher. After a match of length N has been found, the compression searches for a longer match at the next input byte. If a longer match is found, the previous match is truncated to a length of one (thus producing a single literal byte) and the longer match is emitted afterwards. Otherwise, the original match is kept, and the next match search is attempted only N steps later.

The lazy match evaluation is also subject to a runtime parameter. If the current match is long enough, the compression reduces the search for a longer match, thus speeding up the whole process. (See the discussion of improving throughput, below.) If compression ratio is more important than speed, the compression module attempts a complete remote search even if the primary match is already long enough.

c. Alternative to Lazy Evaluation Method

The lazy match evaluation is not performed for the fastest compression modes (Levels 1 to 3). For these fast modes, new strings are inserted in the hash table only when no match is found, or when the match is not too long. This degrades the compression ratio but saves time since there are both fewer insertions and fewer searches.

B. Detailed Description of Compression: Second Stage

Continuing to refer to FIG. 9, a plurality of Huffman trees **460** are used to compress the data to still a further stage. Once this stage has been reached, the data appears as a stream of pointers and literals. In this stage, the compression module looks for repetitions in this stream of pointers and literals. Within the pointers, there may be a match of lengths ("match lengths") or of distances ("match distances"). Literals and match lengths are compressed with a first Huffman tree, and match distances are compressed with a second tree. The compression subroutine deals with a block of the data stream at a time. The trees are stored in a compact form at the start of each block. A block can have any size, except that the compressed data for one block must fit in available memory. The block is terminated when the compression module determines that it would be useful to start another block with fresh trees. (This is somewhat similar to "Compress" and "Zip" programs by Unix.)

The compression module was designed to allow single pass compression without any backwards seek, and without a prior knowledge of the uncompressed input size or the available size on the output media. If the compression method is 0 (stored) however, the stored data comes after the "crc" and "compressed size." This enables fast "decompression."

Compression is always performed, even if the compressed data is slightly larger than the original. This may occur when the data within 32 kilobytes is not compressible or if the compression module has been set to Level 0 to turn off the compression. The worst case expansion is a few bytes for the compression module buffer header, plus 5 bytes every 32K block, or an expansion ratio of 0.015% for large data. The actual number of used disk blocks almost never increases.

C. Decompression

Information needed for the decompression is inserted at the head of the compressed data. The decompression module simply reverses the Huffman tree compression, which results in a sequence of literal bytes and pointers. The decompression module replicates the strings pointed to by the distance/length pointers.

IV. Improving Throughput: Adaptable Compression

FIG. **10** illustrates the process of adaptive file compression.

Several factors can affect the throughput of a file transfer. These factors include varying bandwidth or delay, CPU availability on the sending or receiving processors, and compressibility of the data. The compression is accomplished in such a way as to optimize the average throughput of the file transfer by modifying compression levels for varying conditions.

The variables involved in total transmission time and their relationships is shown by the formula:

EQ. 1: $T = C + M + X + D$

where T is the total time required to transmit a buffer of data, C is the time required to the compression module the data, M is the marshalling time, that is the time required to reorder the data to transmission byte order and to encrypt if desired, X is the time required to transmit the data and D is the time required to decompress the data.

The variables of M and D are small compared to C and X. C and D are both proportional to the amount of data transmitted, but C is much larger than D. X is proportional to the amount of data transmitted and is inversely proportional to the bandwidth of the communication channel. The delay and pipeline size of the communication channels are also significant factors.

For instance, a satellite bandwidth may be 256 kilobytes per second and its delay 1.25 seconds per round trip. When the data is sent and an acknowledgment is received, that is one round trip. The amount of data which may be sent through the pipe at one

time ("pipeline size") may be about 28 kilobytes. When the DCE software is installed, the pipeline size should be set to 28 kilobytes.

For very large data files which may consist of hundreds or thousands of buffers, it is advantageous to attempt to minimize T. The time to compress, C, can be reduced by using a lower level of compression, but this also results in more data to be transmitted, which in turn increases X. The relationship between C and X can be stated as:

$$X = A/C$$

where A is a data dependent factor and depends on the bandwidth and the compression level. Assuming that M and D are negligible, the formula for total time becomes:

$$T = C + A/C$$

Attempting the optimal compression level to minimize T would be very difficult and the time required to determine that value would also increase T. Since the compressibility of the data and the available CPU cycles at both ends if the transmission are also major factors, it is not practicable to determine the optimal compression level for each buffer.

However, one can vary the compression level and see whether the throughput improves or degrades. If throughput improves, one can change the compression level in the same direction as the previous time. If throughput degrades, one can change the compression level in the opposite direction.

For example, if a change in the compression level from 4 to 5 improves throughput, then one can increase the compression level still further to 6. If the compression level of 6 then decreases throughput, the compression level can be set back to 5 for the next attempt.

Continual variation of the compression level is unlikely to achieve optimized throughput for any single transmission. However, continual variation of the compression level is likely to yield a better average throughput for a wide range of files over varying conditions than are achieved with a fixed compression level.

Reffering to FIG. 10, the Level is set **700** to A, preferably a number between 0 and 9. 32 kilobytes of data are compressed **710** at Level A. Throuput T1 is calculated **720** for Level A. Level A is set **730** to A_{new}, while keeping A within Levels o to 9. The next 32 kilobytes are compressewd **740** at Level A_{new} and the throughpput at Level A_{new}, T_{new} is determined**750.** T_{new} is compared to T1 **760.**

If T_{new} is greater than T1, then T1 is set **800** to T_{new} and A_{new} is compared **810** to A. If A_{new} is greater than A, A is set equal **820** to A plus 1. If A_{new} is not greater than A, A is set equal **790** to A minus 1. The steps repeat, starting with step **730**.

If T_{new} is not greater than T1, then T1 is set **800** to T_{new} and A_{new} is compared **780** to A. If A_{new} is less than than A, A is set equal **820** to A plus 1. If A_{new} is not less than A, A is set equal **790** to A minus 1. The steps repeat, starting with step **730**.

V. Encryption, Marshalling and Transmission

Referring to FIG. **6**, FIG. **7A**, and FIG. **7B**, the compressed buffer is put into a queue **540**. The compressed buffer may be encrypted, if desired, and marshalled by an encryption and marshalling module **550** such as Distributed Computing Environment ("DCE") by Open Group. Operating systems often order data either "big endian" or "little endian" (high order byte or low order byte). For example, VMS and Windows NT use little endian; UNIX uses big endian. Marshalling means ordering the data into a platform independent byte order, so that it can be properly received and understood without regard to how the operating system organized it. After marshalling, the buffer is transmitted to the remote file transfer client **440** via a remote procedure call **570**, such as a DCE pipe. The pipe **570** includes a callback mechanism which provides for a continuous read-send loop. That is to say that the primary file transfer server **410** can send a first buffer via the remote procedure call **140** and then read a second buffer of data and send it too, without waiting for the remote file transfer client **440** to receive the first buffer and request another. Other remote procedure calls in alternative embodiments of the invention do not include the callback mechanism.

If there should be an interruption during transmission of the file, the system attempts an automatic recovery. The remote file transfer client **440** will use an automatic callback system three times to attempt to re-establish communications. If communication is successfully re-established, the transfer of data will resume where it left off. This is accomplished by the remote file transfer client **440** knowing how many bytes of data it has and telling the primary file transfer server **410** to pick up at the next byte. If the remote file transfer client **440** has 1000 bytes of data, it will tell the primary file transfer server **410** to begin transmitting at byte number 1001. If the attempts are unsuccessful, i.e. the communications failure is complete, the transfer can be re-started manually, in recover mode, to resume from the point of interruption.

VI. At Remote Location

At the remote location, the transmitted data is de-crypted, unmarshalled, decompressed and written to disk. This may be accomplished in many ways as detailed

in the next section, but is preferably accomplished in one of two ways, depending on whether the data is to be viewed in near real time.

A. Without Near Real Time Viewing

If it is not desired to view the data in near real time, it is preferable to allow the remote file transfer client **440** to write data directly to the sharable data and log format files, instead of using the remote file transfer server to do so. In fact, in such a situation, a remote file transfer server is not required. In addition, in such a case, the remote procedure call preferred is a DCE pipe.

As illustrated in FIG. **6**, in such a case, the transmitted data is received by the remote file transfer client **440**, de-crypted and unmarshalled using a de-cryption/un-marshalling module **580**. The remote file transfer client **440** then decompresses the transmitted data with the decompression module 590 and uses its own copy of the RWW module **490** to write the data to the sharable remote data format file **450** and/or the sharable remote log format file **460** and create the semaphore files, **470, 480**. After the data storage is completed, the data may be rendered into a log and printed or viewed on the remote monitor.

B. With Near Real Time Viewing

If the data is to be viewed in near real time, it is preferable to allow the remote file transfer server to launch the remote renderer and remote monitor. In such a case, it is preferable that the remote file transfer server, and not the remote file transfer client, handle writing the data to the sharable data and log format files. In addition, in such a case, the remote procedure call need not be the DCE pipe, because in real time, the data is accumulating more slowly and may be be transmitted at a slower rate.

As described in co-pending U.S. Patent Application Serial Number 08/772,956 and as illustrated in FIG.**7A** and FIG.**7B**, the transmitted data is received by the remote file transfer client **440**, de-crypted and unmarshalled using a de-cryption/un-marshalling module **580**. The data is then put into a remote queue **600**. Then the remote file transfer client encrypts, if desired, and marshalls the data using an encrytption and marshalling module **610** and uses a second remote procedure call **620** to send the data to the remote file transfer server. The remote file transfer server decrypts and un-marshalls the data using a de-cryption/un-marshalling module and decompresses it using a decompression module **640**. Then using its own copy of the RWW module **490**, the remote file transfer server writes the data to writes the data to the sharable remote data format file and/or the sharable remote log format file, and creates the related semaphore files **470, 480**.

As described in greater detail in co-pending U.S. Patent Application Serial Number 08/772,956, if it is desired to view the data as it is being written, the remote renderer can use its own copy of the RWW module **490** to read the data from the sharable log format file as it is being written and sends the data, through drivers, to the remote printer or remote monitor. This allows a person at the remote location to view or to print the data while it is being written to the remote sharable graphics data format **66**.

The invention described in co-pending U.S. Patent Application Serial Number 08/772,956 also provides a method of person-to-person communication at the same time the log transmission is occurring.

The present invention can be used to transfer any type of file data and is especially useful in transmitting data as it is being acquired.

One benefit of the present invention is that it allows files to be transferred securely, especially while "on the line."

Another benefit of the present invention is that it allows files to be transferred while making maximum use of low bandwidth connections.

Another benefit of the present invention is that it allows a file to be transferred while adaptably compressing the file to improve transmission throughput.

Another benefit of the present invention is that it overcomes the disadvantages of the File Transfer Protocol.

Another benefit of the present invention is that it allows files to be transferred taking into account the unique requirements of mobile network connections.

Another benefit of the present invention is that it allows a file to be transferred as it is compiled in at least near real time from one location to a remote location remote from the primary for viewing or other use.

Another benefit of the present invention is that it allows a well data file to be transferred as it is compiled in at least near real time from a wellsite to a remote location remote from the well site for viewing or other use.

Another benefit of the present invention is that it provides a recovery method should communications be lost.

Another benefit of the present invention is that it sends well data files from a wellsite to a remote location in near real time, in such a way that the data files are not susceptible to being misdirected or lost.

Another benefit of the present invention is that it can maintain the confidentiality of the well data while it is being transmitted.

Another benefit of the present invention is that it allows transferring files in near real time from one location to a remote location so that so that persons can view the files in near real time, without the expense of travelling to the primary location.

Another benefit of the present invention is that it allows transferring well data files in near real time from wellsite to a remote location remote from the wellsite so that persons can view the well data files near real time as they are being compiled, without the expense of travelling to the wellsite and without being exposed to the hazards of the wellsite.

The principles, preferred embodiments, and modes of operation of the present invention have been described in the foregoing specification. The invention is not to be construed as limited to the particular forms disclosed, because these are regarded as illustrative rather than restrictive. Moreover, variations and changes may be made by those skilled in the art without departing from the spirit of the invention.

What is claimed is:

CLAIMS

1. A system for handling and transmitting a file having a degree of compressibility over a communication channel having a physical bandwidth wherein the improvement comprises;

a.) a file transfer server at a first location;

b.) a file transfer client at a second location; and

c.) a means for compressing the file based on the physical bandwidth, the capabilities of the transmitting and receiving processors and the degree of compressibility of the file.

2. A system for managing and transmitting a file having a degree of compressibility and a plurality of buffers over a communication channel having a physical bandwidth wherein the improvement comprises:

a.) a file transfer server at a first location;

b.) a file transfer client at a second location; and

c.) a feedback loop for optimally compressing the file for transmission, including:

1.) means for compressing a first buffer to a first level of compressibility;

2.) means for evaluating the efficiency of the compression of the first buffer;

3.) means for adjusting the compression of second buffer based on the evaluation of the compression of the first buffer.

3. A system for managing and transmitting a file having a plurality of buffers over a communication channel wherein the improvement comprises:

a.) a means for compressing buffers of the file in a stream in real time while the file is being written.

4. A system for managing and transmitting a file over a communication channel wherein the improvement comprises:

a.) a transmission of the file having a state and a location; and

b.) a means for maintaining the state and location of the transmission within the file so that transmission can be resumed in the event of an interruption at the point of the interruption.

5. A system as in claim 4 wherein the interrupted transmission can be automatically resumed.

6. A system as in claim 4 wherein the interrupted transmission can be manually resumed.

7. A system for managing and transmitting a file over a communication channel wherein the improvement comprises:

a.) a means for encrypting the file in a stream before and during transmission; and

b.) means for de-crypting the file after receipt of transmission.

8. A system for managing and transmitting a file having a degree of compressibility and a plurality of buffers over a communication channel having a physical bandwidth wherein the improvement comprises:

a.) a transmitting processor;

b.) a receiving processor;

c.) a means for optimally compressing the file based on the physical bandwidth, the capabilities of the transmitting and receiving processors and the degree of compressibility of the file;

d.) a means for compressing the buffers of the file in a stream in real time while the file is being written;

e.) a transmission of the file having a state and a location;

f.) a means for maintaining the state and location of the transmission within the file so that transmission can be resumed in the event of an interruption at the point of the interruption;

g.) a means for encrypting the file in a stream before and during transmission; and

h.) means for de-crypting the file after receipt of transmission.

9. A method of compressing a source file, having a plurality of buffers, for transmission over a communication channel, comprising:

a.) selecting a first buffer of the source file;

b.) compressing the first buffer to a first compression level;

c.) marshalling the first buffer;

d.) transmitting the first buffer;

e.) de-compressing the first buffer;

f.) writing the first buffer to a destination file;

g.) determining a first throughput which was used for steps (a) through (f);

h.) selecting the compression level of a second buffer based on the first throughput; and

i.) repeating steps (a) through (h) for each of the buffers in turn until all of the buffers of the source file have been transmitted and have been written to the destination file.

10. A method as in claim 9, wherein the marshalling steps include encrypting the buffers and the de-compressing steps include de-encrypting the buffers.

11. A method as in claim 10, further comprising the step of writing to the source file while performing one or more of steps (a) through (i).

12. A system for handling and transmitting a first file over a communication channel comprising:

a.) a means for reading while writing data to the first file;

b.) a means for compressing the first file;

c.) a means for queueing the first file;

d.) a means for marshalling the first file;

e.) a means for transmitting the first file from a first location to a second location;

f.) a means for unmarshalling the first file;

g.) a means for decompressing the first file; and

h.) a means for writing the first file to a second file.

13. A system for handling and transmitting a first file as in claim 12 wherein the means for accomplishing steps a through d comprises a file transfer server.

14. A system for handling and transmitting a first file as in claim 12 wherein the means for accomplishing steps f through g comprises a file transfer client.

15. A system for handling and transmitting a first file as in claim 13 wherein the file transfer server includes a read while write module.

16. A system for handling and transmitting a first file as in claim 13 wherein the file transfer server includes a compression module.

17. A system for handling and transmitting a first file as in claim 16 wherein the compression module uses a deflation algorithm.

18. A system for handling and transmitting a first file as in claim 16 wherein the compression module uses a Huffman tree.

19. A system for handling and transmitting a first file as in claim 16 wherein the compression module compresses in a first stage to produce literals and pointers.

20. A system for handling and transmitting a first file as in claim 19 wherein the compression module compresses in a second stage to produce a Huffman tree and a block.

21. A system for handling and transmitting a first file as in claim 12, further comprising

a means for encrypting the first file before marshalling the first file and a means for decrypting the first file after the first file is transmitted.

22. A system for handling and transmitting a first file as in claim 14 wherein the file transfer client includes a read while write module.

23. A system for handling and transmitting a first file as in claim 21, wherein the first file is a sharable file.

24. A system for handling and transmitting a first file over a communication channel comprising:

a.) a means for reading while writing data to the first file;

b.) a means for compressing the first file;

c.) a means for queueing the first file;

d.) a means for marshalling the first file;

e.) a means for transmitting the first file from a first location to a second location;

f.) a first means for unmarshalling the first file;

g.) a means for re-queueing the first file;

h.) a means for re-marshalling the first file;

i.) a remote procedure call for sending the first file within the second location;

j.) a second means for unmarshalling the first file;

k.) a means for decompressing the first file; and

l.) a means for writing the first file to a second file.

25. A system for handling and transmitting a first file as in claim 24 wherein the means for accomplishing steps a through d comprises a first file transfer server.

26. A system for handling and transmitting a first file as in claim 24 wherein the means for accomplishing steps f through g comprises a file transfer client.

27. A system for handling and transmitting a first file as in claim 25 wherein the file transfer server includes a read while write module.

28. A system for handling and transmitting a first file as in claim 25 wherein the file transfer server includes a compression module.

29. A system for handling and transmitting a first file as in claim 28 wherein the compression module uses a deflation algorithm.

30. A system for handling and transmitting a first file as in claim 28 wherein the compression module uses a Huffman tree.

31. A system for handling and transmitting a first file as in claim 28 wherein the compression module compresses in a first stage to produce literals and pointers.

32. A system for handling and transmitting a first file as in claim 31 wherein the compression module compresses in a second stage to produce a Huffman tree and a block.

33. A system for handling and transmitting a first file as in claim 24, further comprising:

a first means for encrypting the first file before marshalling the first file; and

a first means for decrypting the first file after the first file is transmitted.

34. A system for handling and transmitting a first file as in claim 33, further comprising:

a second means for encrypting the first file after the first file is transmitted;

a first means for decrypting the first file after the first file has been sent through the remote procedure call within the second location.

35. A system for handling and transmitting a first file as in claim 24 wherein the means for accomplishing steps j through l comprises a second file transfer server.

36. A system for handling and transmitting a first file as in claim 14 wherein the file transfer client includes a read while write module.

37. A system for handling and transmitting a first file as in claim 35, wherein the first file is a sharable file.

38. A method for handling and transmitting a first file over a communication channel comprising the steps of:

a.) reading while writing data to the first file;

b.) compressing the first file;

c.) queueing the first file;

d.) marshalling the first file;

e.) transmitting the first file from a first location to a second location;

f.) unmarshalling the first file;

g.) decompressing the first file; and

h.) writing the first file to a second file.

39. A method for handling and transmitting a first file as in claim 38 wherein steps a through d are accomplished by a file transfer server.

40. A method for handling and transmitting a first file as in claim 38wherein steps f through g are accomplished by a file transfer client.

41. A method for handling and transmitting a first file as in claim 39 wherein the file transfer server includes a read while write module.

42. A method for handling and transmitting a first file as in claim 38 wherein the file transfer server includes a compression module.

43. A method for handling and transmitting a first file as in claim 38 wherein the compression step includes use of a deflation algorithm for a first stage of compression.

44. A method for handling and transmitting a first file as in claim 38 wherein the compression step uses a Huffman tree for a second stage of compression.

45. A method for handling and transmitting a first file as in claim 43 wherein the use of the deflation algorithm in the first stage produces literals and pointers.

46. A method for handling and transmitting a first file as in claim 44 wherein the use in a second stage of the Huffman tree produces a Huffman tree and a block.

47. A method for handling and transmitting a first file as in claim 38, further comprising the steps of:

encrypting the first file before marshalling the first file; and

decrypting the first file after the first file is transmitted.

48. A system for handling and transmitting a first file as in claim 40 wherein the file transfer client includes a read while write module.

49. A system for handling and transmitting a first file as in claim 38, wherein the first file is a sharable file.

50. A system for handling and transmitting a first file as in claim 38, wherein the second file is a sharable file.

51. A method for handling and transmitting a first file over a communication channel comprising the steps of:

a.) reading while writing data to the first file;

b.) compressing the first file;

c.) queueing the first file;

d.) marshalling the first file;

e.) transmitting the first file from a first location to a second location;

f.) unmarshalling the first file;

g.) re-queueing the first file;

h.) re-marshalling the first file;

i.) sending via a remote procedure call the first file within the second location;

j.) unmarshalling the first file a second time;

k.) decompressing the first file; and

l.) writing the first file to a second file.

52. A method for handling and transmitting a first file as in claim 51 wherein the steps a through d are accomplished by a first file transfer server.

53. A method for handling and transmitting a first file as in claim 51, wherein steps f through g are accomplished by a file transfer client.

54. A method for handling and transmitting a first file as in claim 52 wherein the file transfer server includes a read while write module.

55. A method for handling and transmitting a first file as in claim 52 wherein the file transfer server includes a compression module.

56. A method for handling and transmitting a first file as in claim 52 wherein the compression step includes use of a deflation algorithm for a first stage of compression.

57. A method for handling and transmitting a first file as in claim 56 wherein the compression step uses a Huffman tree for a second stage of compression.

58. A method for handling and transmitting a first file as in claim 57 wherein the use of the Huffman tree produces a Huffman tree and a block.

59. A system for handling and transmitting a first file as in claim 56 wherein the use of the deflation algorithm in the first stage produces literals and pointers.

60. A method for handling and transmitting a first file as in claim 51, further comprising the steps of:

encrypting the first file before marshalling the first file; and

decrypting the first file after the first file is transmitted to the second location.

61. A system for handling and transmitting a first file as in claim 60, further comprising:

encrypting the first file a second time, after the first file is transmitted; and

decrypting the first file after the first file has been sent through the remote procedure call within the second location.

62. A system for handling and transmitting a first file as in claim 51, wherein steps j through l are accomplished using a second file transfer server.

63. A system for handling and transmitting a first file as in claim 53, wherein the file transfer client includes a read while write module.

64. A system for handling and transmitting a first file as in claim 61, wherein the first file is a sharable file.

65. A method of transmitting a file over a communication system using TCP/IP protocol comprising transmitting the file wherein the improvement comprises:

recovering the file in the event of a communication interruption.

66. A system of transmitting a file over a communication system using TCP/IP protocol wherein the improvement comprises:

a means for recovering the file in the event of a communication interruption.

AMENDED CLAIMS

[received by the International Bureau on 22 July 1998 (22.07.98); original claims 65 and 66 amended; new claims 67-72 added; remaining claims unchanged (3 pages)]

63. A system for handling and transmitting a first file as in claim 53, wherein the file transfer client includes a read while write module.

64. A system for handling and transmitting a first file as in claim 61, wherein the first file is a sharable file.

65. A system adapted for receiving a data file at a first location and for transmitting said data file from said first location to a second location via a communication link, comprising:

a first apparatus at said first location adapted for receiving the data file and for modifying said data file thereby generating a modified data file;

a sharable file at said first location, said sharable file including a semaphore file;

a read-while-write apparatus at said first location operatively interconnected between the first apparatus and the sharable file for reading the modified data file from the first apparatus while writing the modified data file to the sharable file;

a primary file transfer server apparatus including a data compressor operatively connected to said sharable file and said semaphore file, said primary file transfer server apparatus being adapted for determining if said semaphore file is associated with said sharable file and, when said semaphore file is associated with said sharable file, for reading said modified data file from said sharable file, while said modified data file is being written to said sharable file by said read-while-write apparatus, while writing said modified data file to said data compressor,

said data compressor of said primary file transfer server apparatus receiving said modified data file from said sharable file, compressing said modified data file, and providing the compressed modified data file to said communication link, the compressed modified data file being transmitted to said second location via said communication link.

66. The system of claim 65, wherein said data compressor comprises an adaptable data compressor adapted for performing adaptable compression of said modified data file.

67. The system of claim 66, wherein said adaptable data compressor has a compression level and a throughput, and wherein, during said adaptable compression, said adaptable data compressor receives said throughput and changes said compression level of said data compressor in accordance with the received throughput.

68. The system of claim 65, wherein said data compressor stores a deflation algorithm and executes the deflation algorithm during the compression of said modified data file.

69. The system of claim 68, wherein said data compressor uses a Huffman tree after the execution of said deflation algorithm.

70. The system of claim 67, wherein said primary file transfer server apparatus further includes an encryption marshalling module connected to said data compressor and adapted for encrypting said compressed modified data file and marshalling the encrypted compressed modified data file prior to transmitting the data file to said second location via said communication link.

71. The system of claim 70, further comprising a decryption and un-marshalling module at said second location and operatively responsive to said data file transmitted to said second location via said communication link, a decompression module operatively connected to said decryption and un-marshalling module, a remote sharable file, and a further read-while-write apparatus operatively interconnected between said decompression module and said remote sharable file.

72. A method of transmitting a data file from a first location to a second location, comprising the steps of:

receiving said data file in a first apparatus located at said first location;

reading said data file from said first apparatus while writing said data file to a sharable file;

reading said data file from said sharable file while writing said data file to a data compressor;

setting a compression level in said data compressor in accordance with a throughput of said data compressor and compressing said data file at said compression level thereby generating a compressed data file;

encrypting and marshalling said compressed data file; and

transmitting the encrypted and marshalled and compressed data file to said second location.

STATEMENT UNDER ARTICLE 19

The claims of this application have been amended by replacing originally filed claims 65 and 66 with new replacement claims 65 and 66, and by adding new claims 67 through 72. The originally filed claims 1 through 64 remain unchanged.

The specification of this application discloses the following general concept:

A data file is received in a first location and is transmitted from the first location to a second location via a communication link. A receiving apparatus (340, 350 in figure 5A), located at the first location, receives the data file, and a first read-while-write module (360 in figure 5A) at the first location will read the data file from the receiving apparatus while writing the data file to a sharable file (370, 380). A further read-while-write apparatus (360 in figure 7A) at the first location reads the data file from the sharable file, while it is being written to the sharable file by the first read while write apparatus (360), and writes the data file to a compression module (530 in figures 6 and 7A). The compression module may include an "adaptable compression" feature (figure 10) wherein the compression level of the compression module varies depending on the throughput of the compression module; alternatively, the compression module may execute a deflation algorithm and it may use a Huffman tree. The data file is compressed in the compression module, and the compressed data file is encrypted and marshalled via the encryption module (550 in figure 6, 7A) prior to being transmitted from the first location to the second location via the communication link.

No "new matter" is being introduced into new claims 65 through 72. As a result, the originally filed specification and drawings of this application remain unchanged.

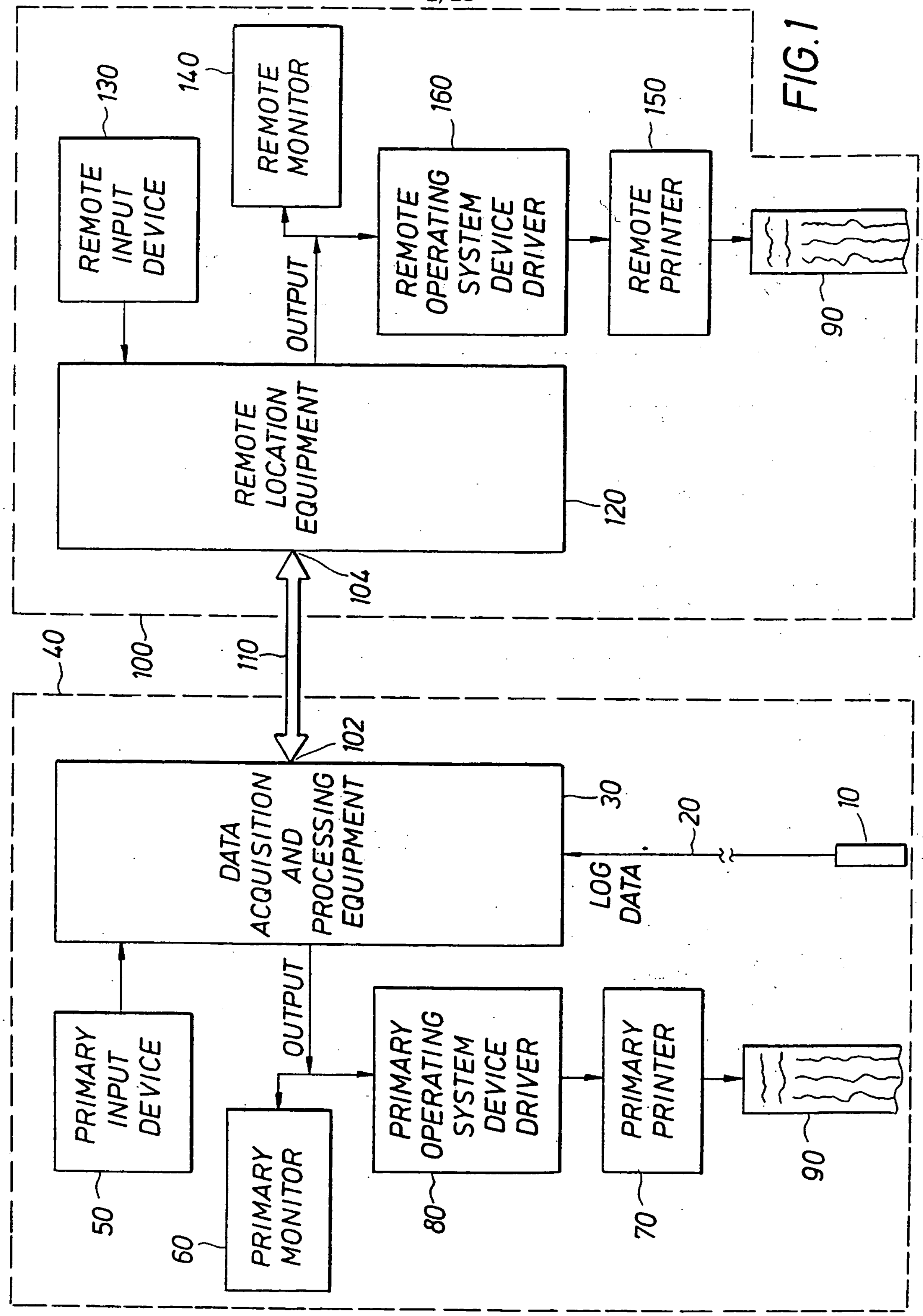

REMOTE INPUT DEVICE
130
REMOTE MONITOR
140
REMOTE OPERATING SYSTEM DEVICE DRIVER
160
REMOTE PRINTER
150
90
OUTPUT
REMOTE LOCATION EQUIPMENT
120
104
40
100
110
102
DATA ACQUISITION AND PROCESSING EQUIPMENT
30
20
10
LOG DATA
OUTPUT
PRIMARY INPUT DEVICE
50
PRIMARY MONITOR
60
PRIMARY OPERATING SYSTEM DEVICE DRIVER
80
PRIMARY PRINTER
70
90

FIG.1

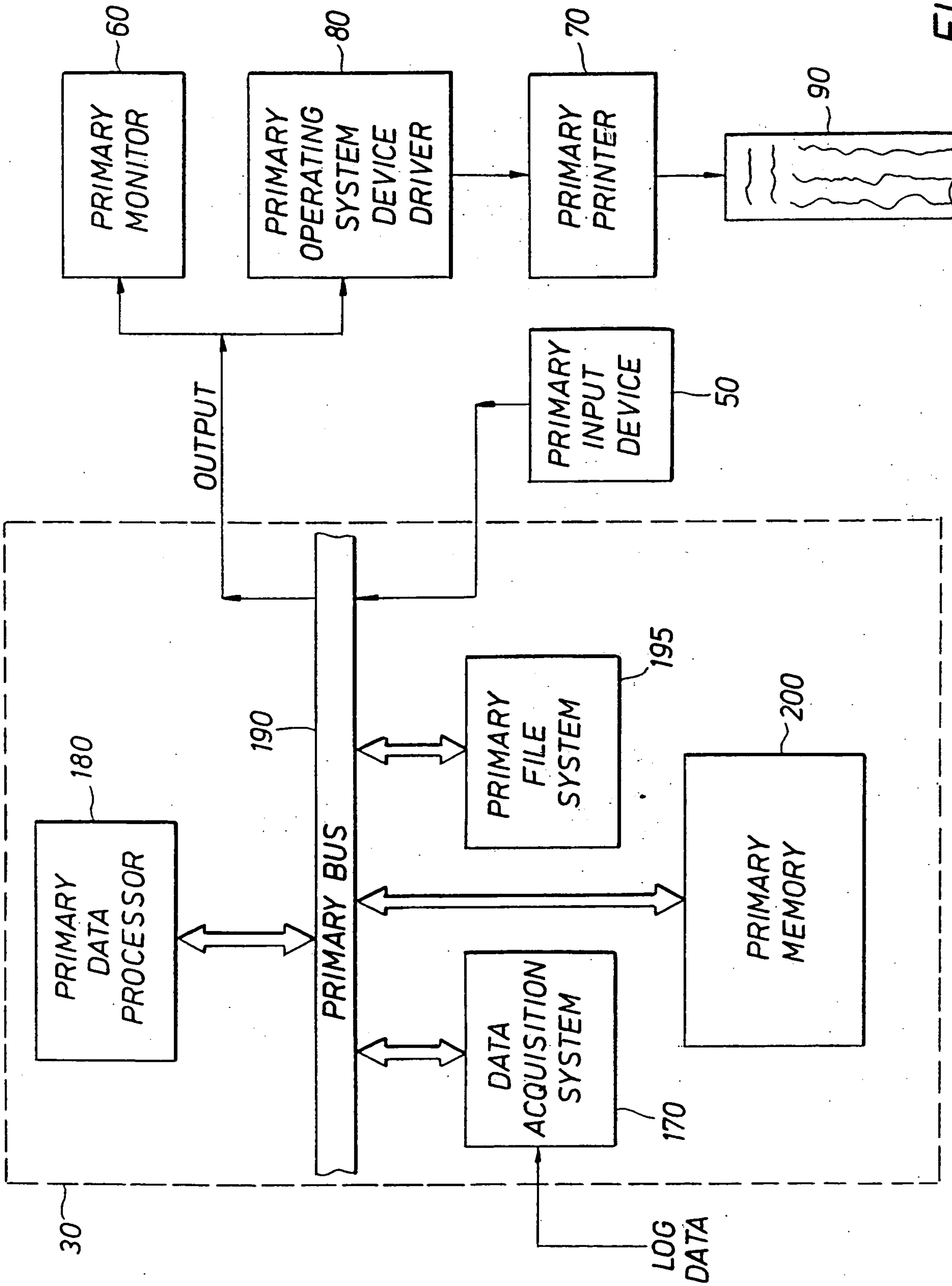

PRIMARY MONITOR
60
PRIMARY OPERATING SYSTEM DEVICE DRIVER
80
PRIMARY PRINTER
70
90
OUTPUT
PRIMARY INPUT DEVICE
50
PRIMARY DATA PROCESSOR
180
PRIMARY BUS
190
PRIMARY FILE SYSTEM
195
PRIMARY MEMORY
200
DATA ACQUISITION SYSTEM
170
30
LOG DATA

FIG. 2

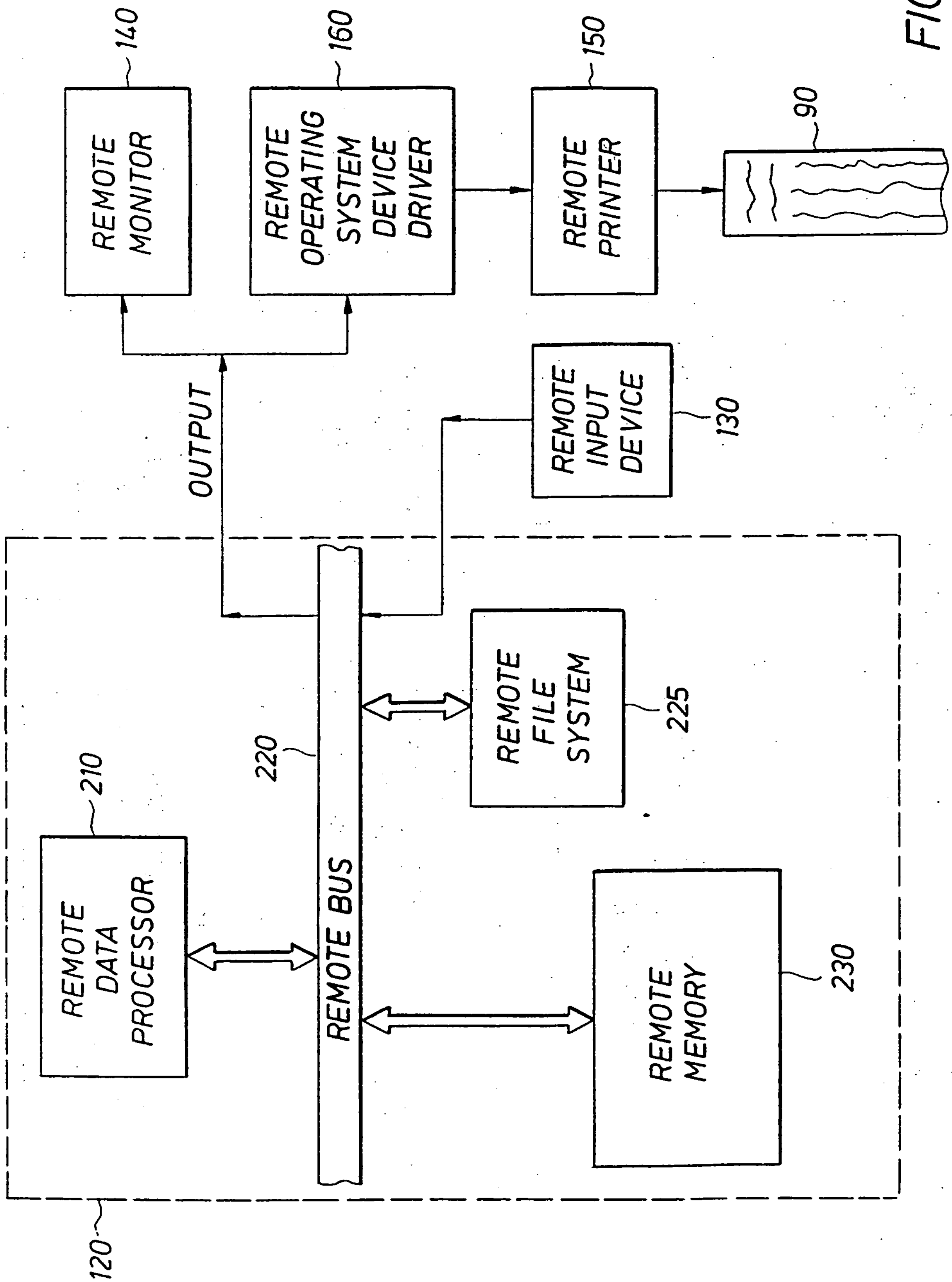

REMOTE MONITOR
140
REMOTE OPERATING SYSTEM DEVICE DRIVER
160
REMOTE PRINTER
150
90
OUTPUT
REMOTE INPUT DEVICE
130
REMOTE DATA PROCESSOR
210
REMOTE BUS
220
REMOTE FILE SYSTEM
225
REMOTE MEMORY
230
120

FIG. 3

FIG. 4A

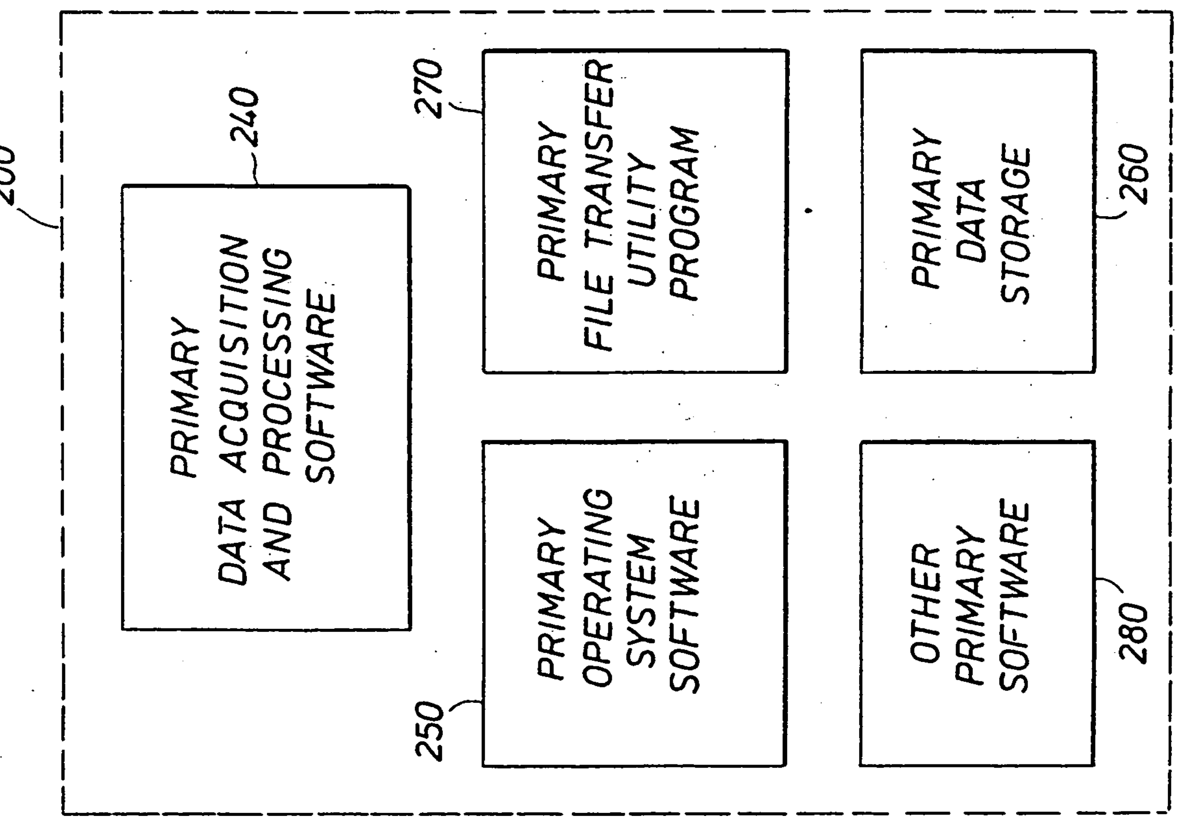

200
240
PRIMARY DATA ACQUISITION AND PROCESSING SOFTWARE
270
PRIMARY FILE TRANSFER UTILITY PROGRAM
250
PRIMARY OPERATING SYSTEM SOFTWARE
PRIMARY DATA STORAGE
260
OTHER PRIMARY SOFTWARE
280

FIG. 4B

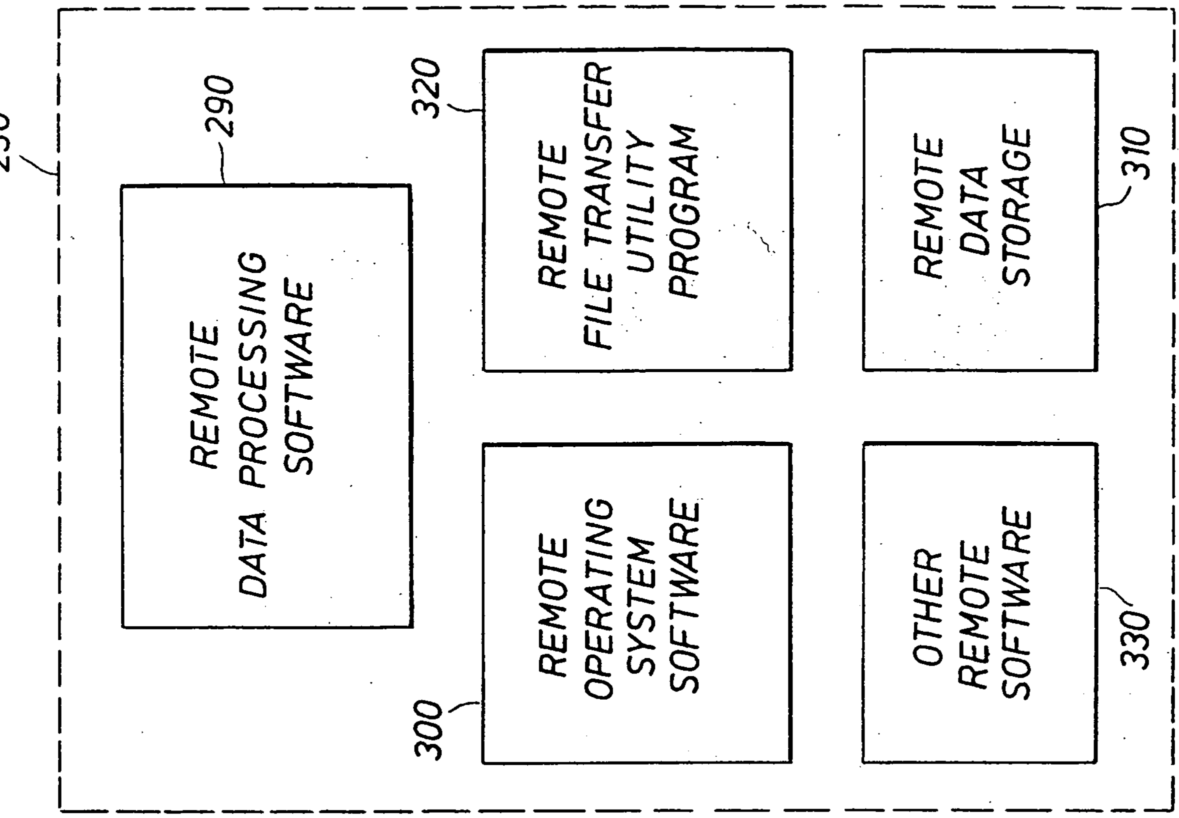

230
290
REMOTE DATA PROCESSING SOFTWARE
320
REMOTE FILE TRANSFER UTILITY PROGRAM
300
REMOTE OPERATING SYSTEM SOFTWARE
REMOTE DATA STORAGE
310
OTHER REMOTE SOFTWARE
330

FIG.5A

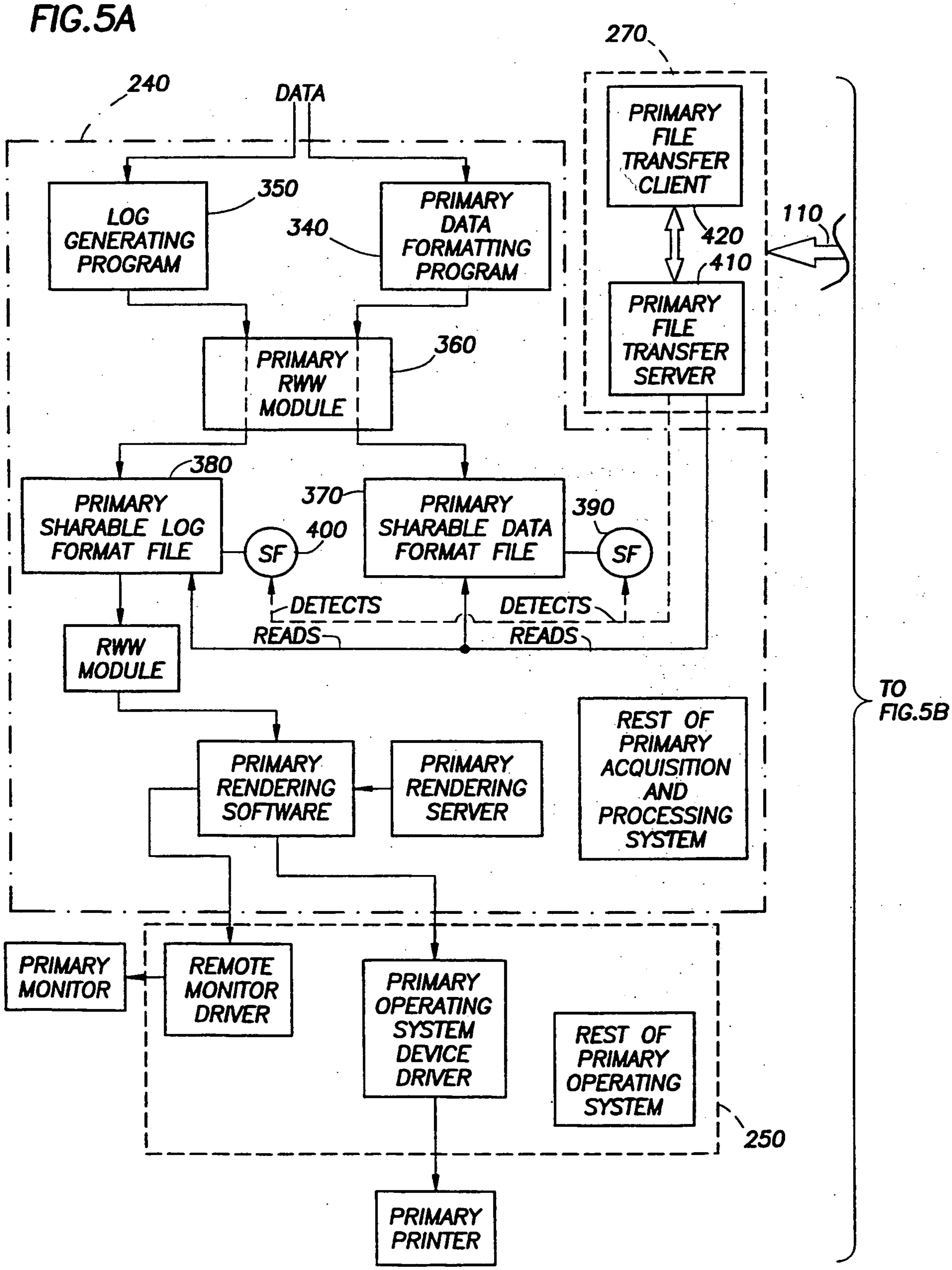
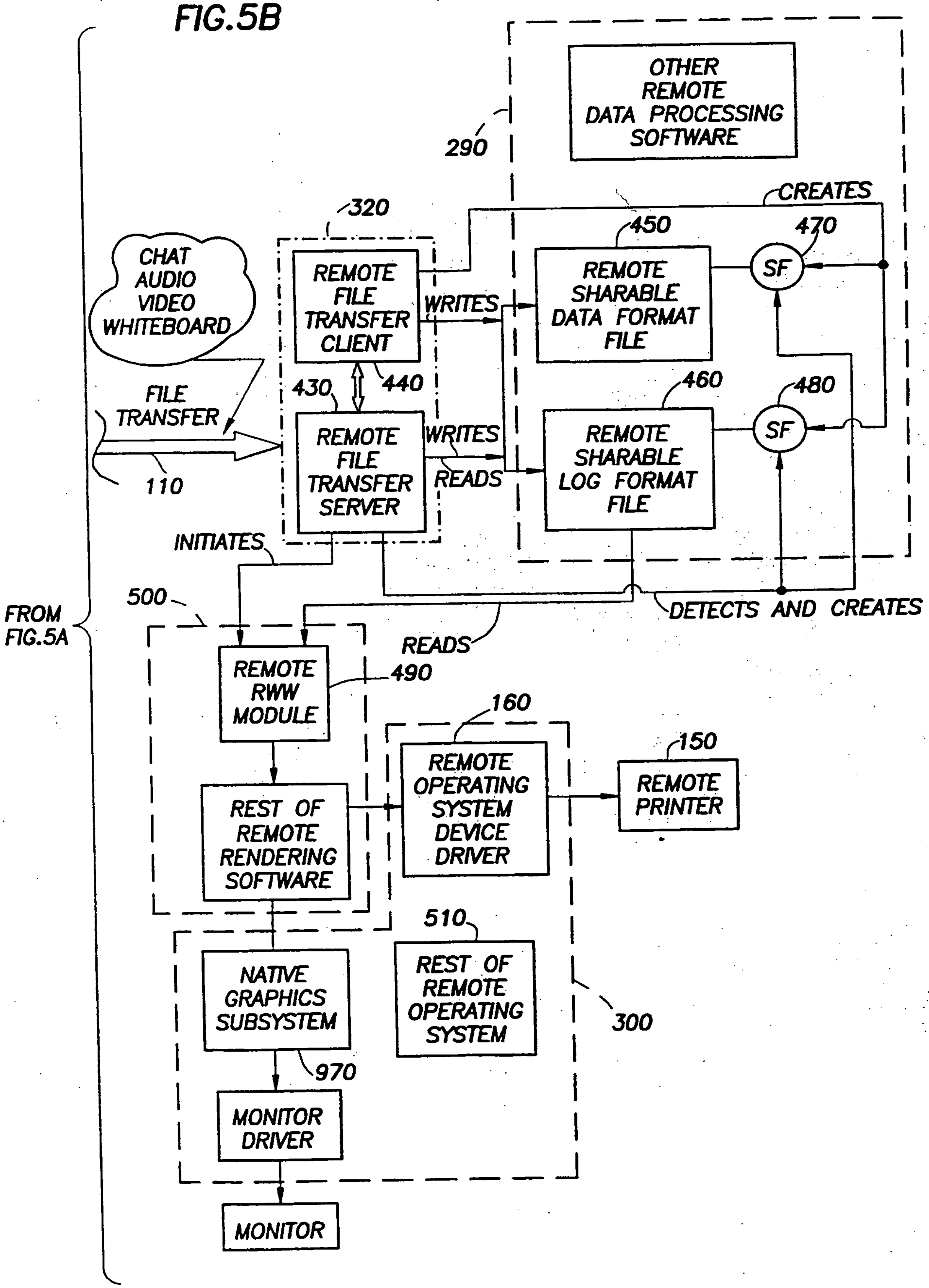
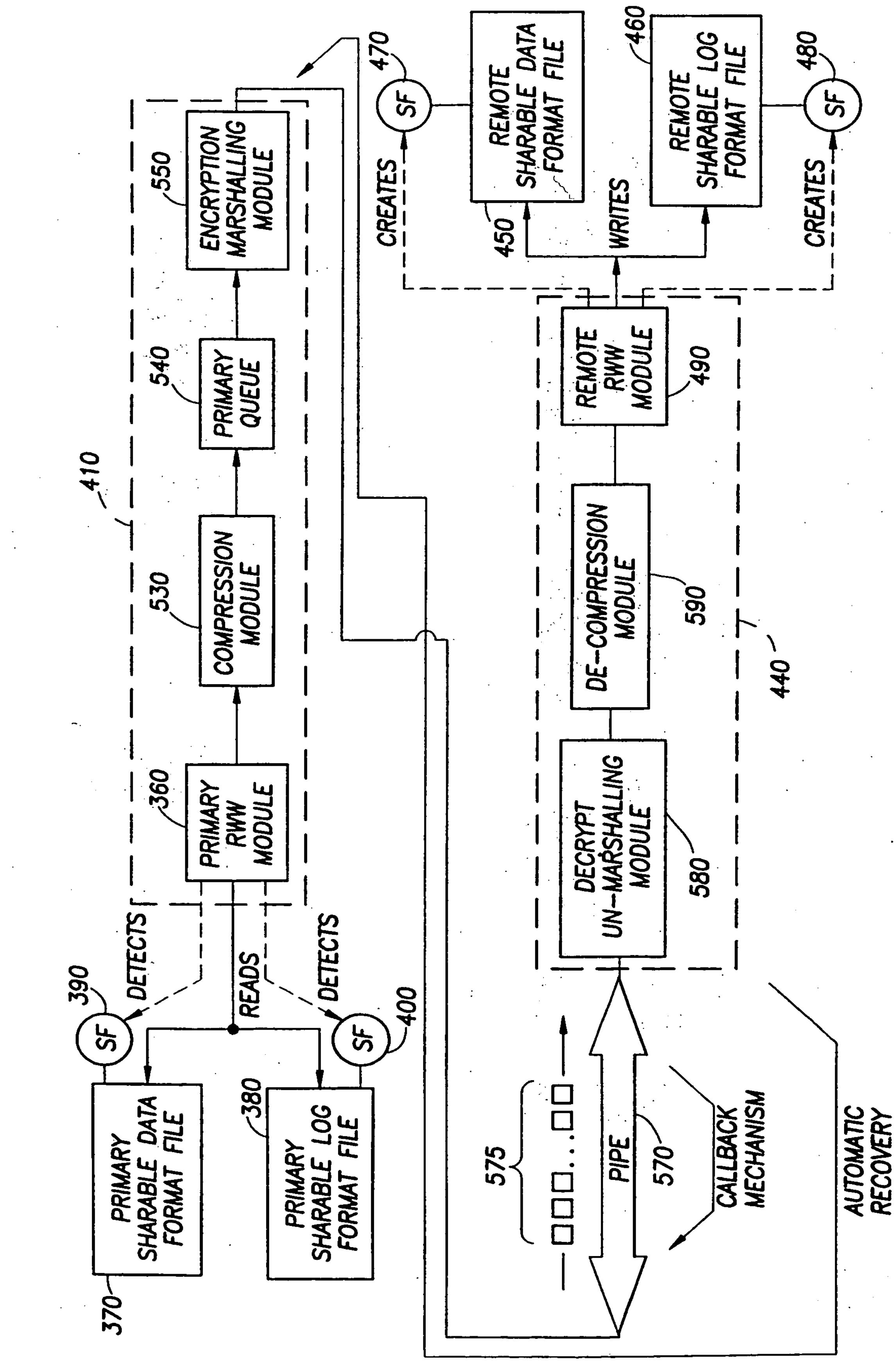

270
240
DATA
PRIMARY FILE TRANSFER CLIENT
LOG GENERATING PROGRAM
350
340
PRIMARY DATA FORMATTING PROGRAM
420
410
110
PRIMARY FILE TRANSFER SERVER
PRIMARY RWW MODULE
360
380
370
PRIMARY SHARABLE LOG FORMAT FILE
SF
400
PRIMARY SHARABLE DATA FORMAT FILE
390
SF
DETECTS
DETECTS
READS
READS
RWW MODULE
TO FIG.5B
REST OF PRIMARY ACQUISITION AND PROCESSING SYSTEM
PRIMARY RENDERING SOFTWARE
PRIMARY RENDERING SERVER
PRIMARY MONITOR
REMOTE MONITOR DRIVER
PRIMARY OPERATING SYSTEM DEVICE DRIVER
REST OF PRIMARY OPERATING SYSTEM
250
PRIMARY PRINTER

FIG.5B



OTHER REMOTE DATA PROCESSING SOFTWARE
290
320
CREATES
450
470
CHAT AUDIO VIDEO WHITEBOARD
REMOTE FILE TRANSFER CLIENT
WRITES
REMOTE SHARABLE DATA FORMAT FILE
SF
430
440
460
480
FILE TRANSFER
REMOTE FILE TRANSFER SERVER
WRITES
READS
REMOTE SHARABLE LOG FORMAT FILE
SF
110
INITIATES
500
DETECTS AND CREATES
FROM FIG.5A
READS
REMOTE RWW MODULE
490
160
150
REST OF REMOTE RENDERING SOFTWARE
REMOTE OPERATING SYSTEM DEVICE DRIVER
REMOTE PRINTER
510
NATIVE GRAPHICS SUBSYSTEM
REST OF REMOTE OPERATING SYSTEM
300
970
MONITOR DRIVER
MONITOR

FIG.6


PRIMARY SHARABLE DATA FORMAT FILE
370
PRIMARY SHARABLE LOG FORMAT FILE
380
SF
390
SF
400
DETECTS
READS
DETECTS
PRIMARY RWW MODULE
360
COMPRESSION MODULE
530
PRIMARY QUEUE
540
ENCRYPTION MARSHALLING MODULE
550
410
575
PIPE
570
CALLBACK MECHANISM
AUTOMATIC RECOVERY
DECRYPT UN-MARSHALLING MODULE
580
DE-COMPRESSION MODULE
590
440
REMOTE RWW MODULE
490
CREATES
WRITES
CREATES
SF
470
REMOTE SHARABLE DATA FORMAT FILE
450
REMOTE SHARABLE LOG FORMAT FILE
460
SF
480

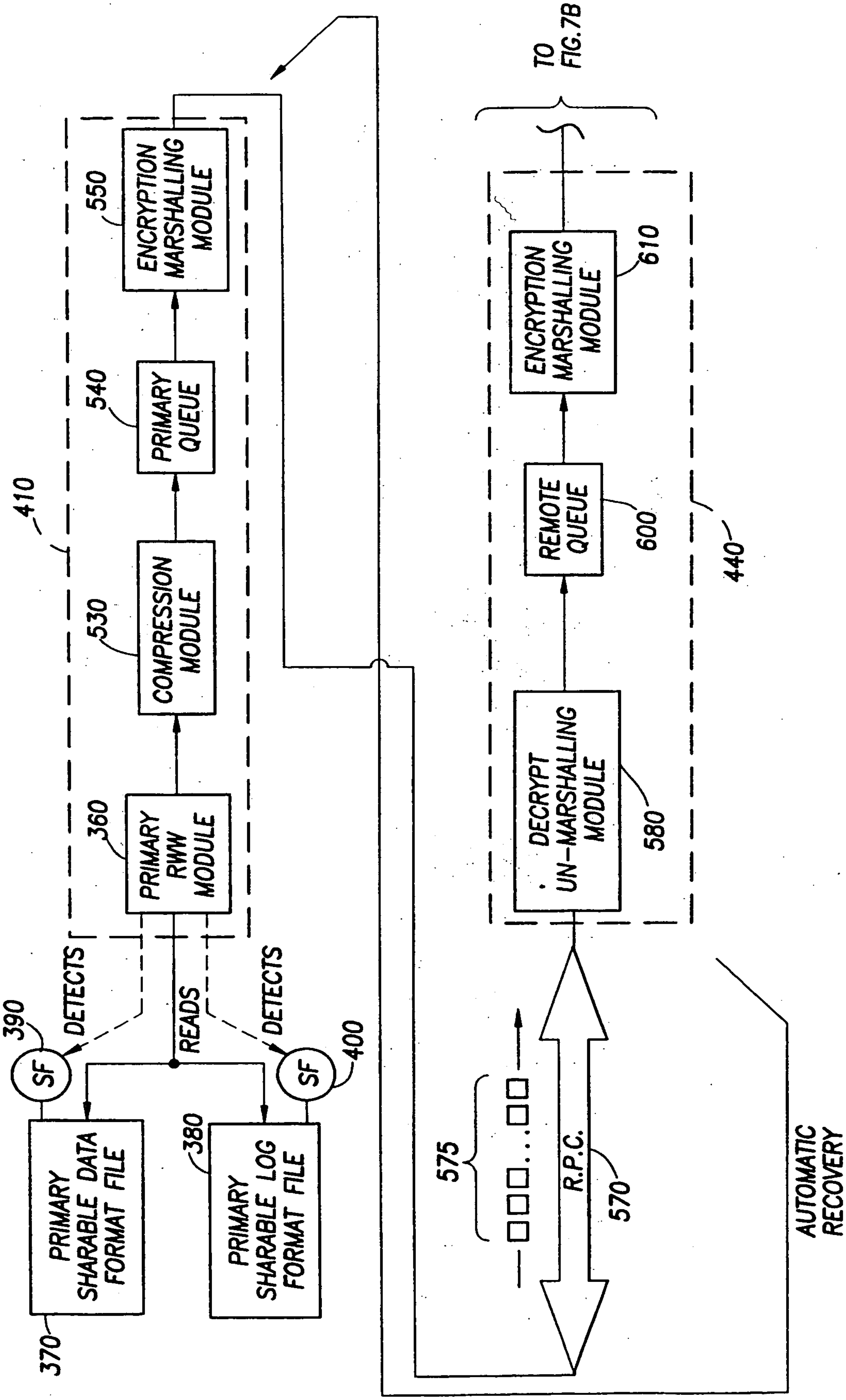

ENCRYPTION MARSHALLING MODULE
550
PRIMARY QUEUE
540
COMPRESSION MODULE
530
410
PRIMARY RWW MODULE
360
DETECTS
READS
DETECTS
SF
390
SF
400
PRIMARY SHARABLE DATA FORMAT FILE
370
PRIMARY SHARABLE LOG FORMAT FILE
380
TO FIG.7B
ENCRYPTION MARSHALLING MODULE
610
REMOTE QUEUE
600
440
DECRYPT UN-MARSHALLING MODULE
580
575
R.P.C.
570
AUTOMATIC RECOVERY

FIG.7A

FIG. 7B

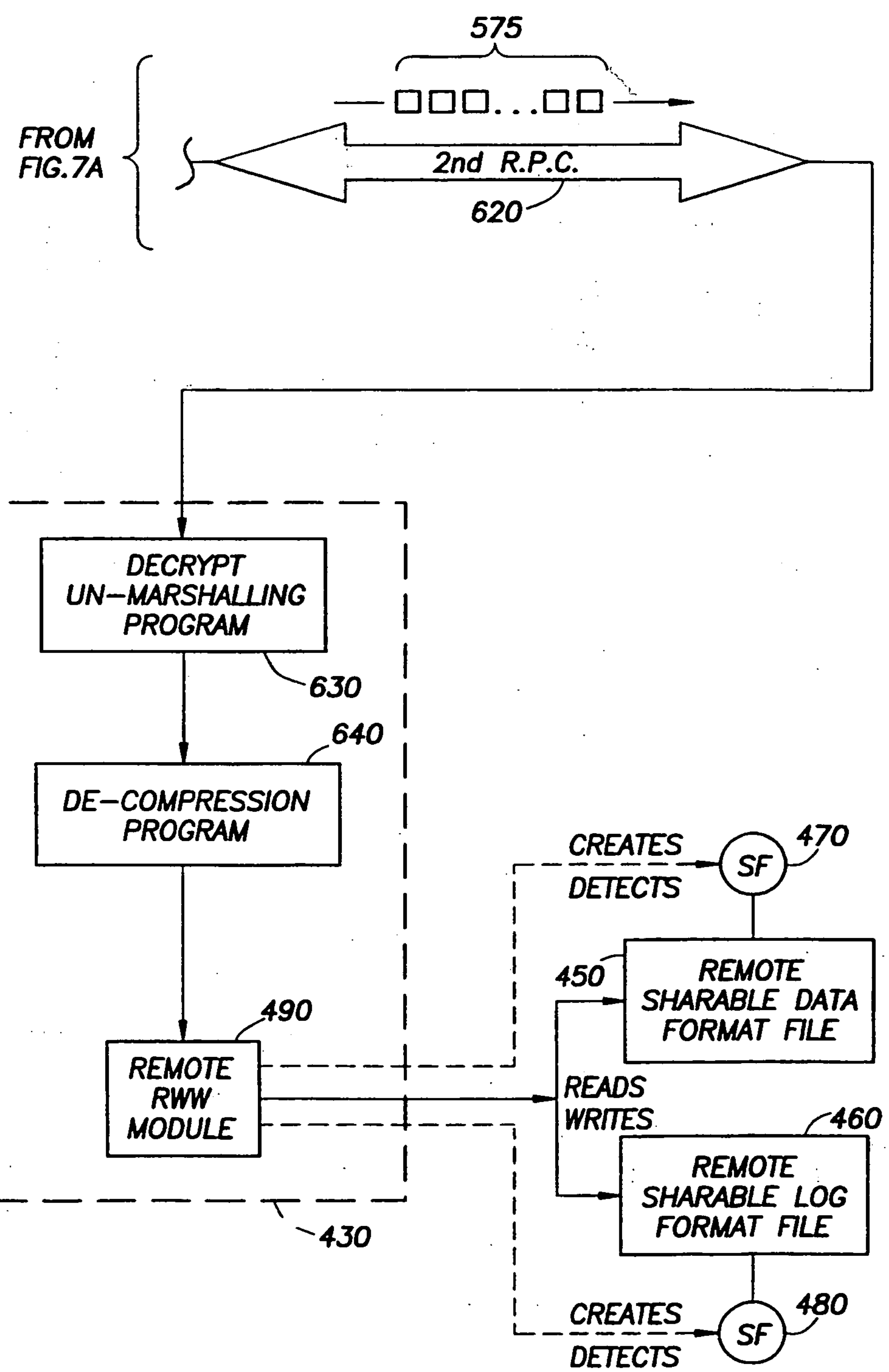

575
FROM FIG. 7A
2nd R.P.C.
620
DECRYPT UN-MARSHALLING PROGRAM
630
640
DE-COMPRESSION PROGRAM
490
REMOTE RWW MODULE
430
CREATES
DETECTS
SF
470
450
REMOTE SHARABLE DATA FORMAT FILE
READS
WRITES
460
REMOTE SHARABLE LOG FORMAT FILE
CREATES
DETECTS
SF
480

FIG. 9

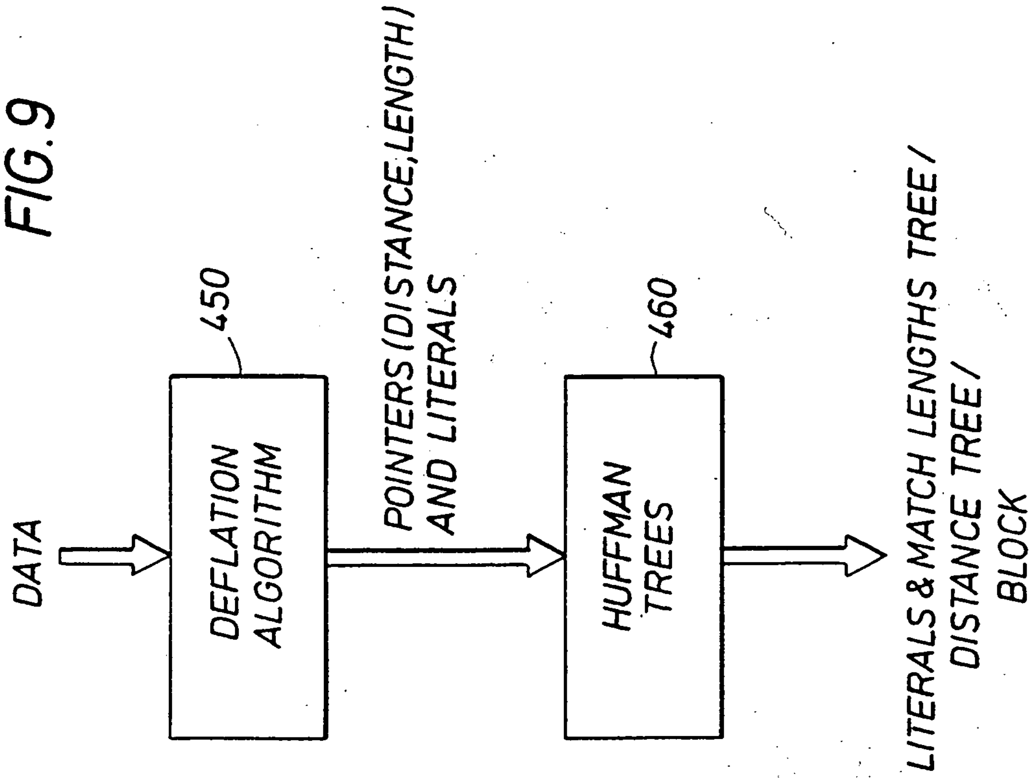

DATA
450
DEFLATION ALGORITHM
POINTERS (DISTANCE, LENGTH) AND LITERALS
460
HUFFMAN TREES
LITERALS & MATCH LENGTHS TREE / DISTANCE TREE / BLOCK

FIG. 8

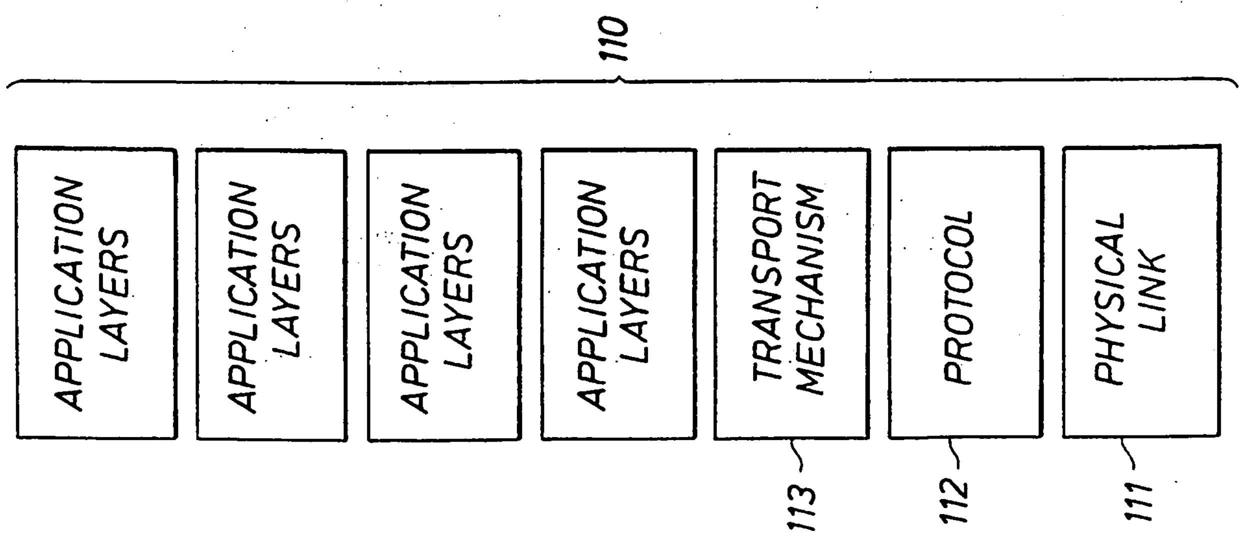

110
APPLICATION LAYERS
APPLICATION LAYERS
APPLICATION LAYERS
APPLICATION LAYERS
TRANSPORT MECHANISM
113
PROTOCOL
112
PHYSICAL LINK
111

FIG. 10

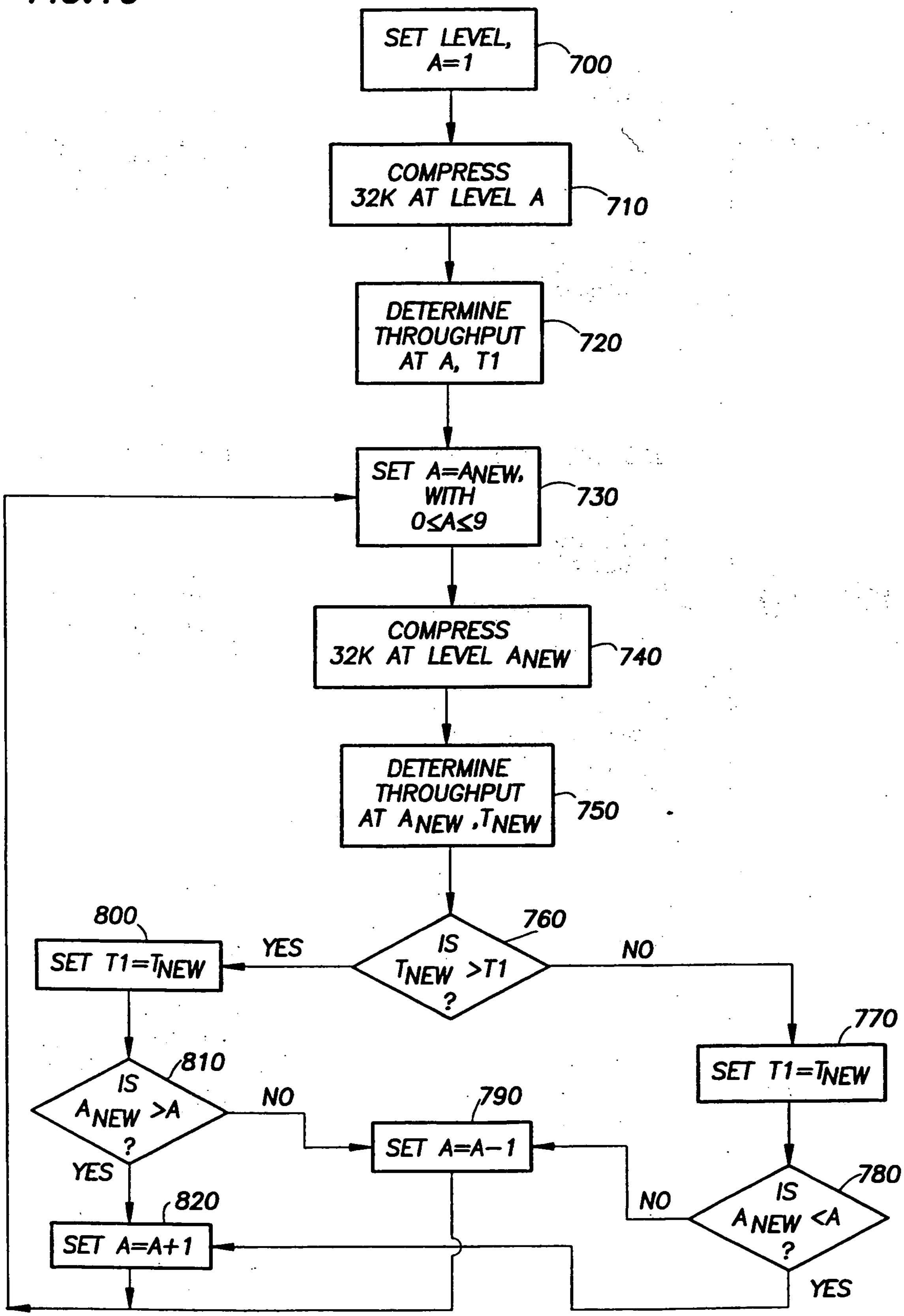

SET LEVEL, A=1
700
COMPRESS 32K AT LEVEL A
710
DETERMINE THROUGHPUT AT A, T1
720
SET A=ANEW, WITH 0≤A≤9
730
COMPRESS 32K AT LEVEL ANEW
740
DETERMINE THROUGHPUT AT ANEW ,TNEW
750
IS TNEW >T1 ?
760
YES
NO
SET T1=TNEW
800
SET T1=TNEW
770
IS ANEW >A ?
810
NO
YES
SET A=A−1
790
IS ANEW <A ?
780
NO
YES
SET A=A+1
820

FIG.11A

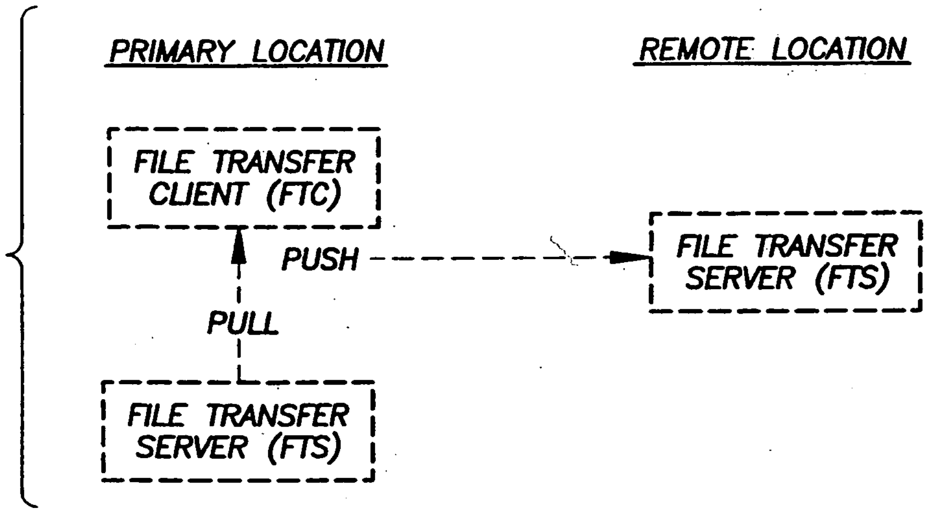


FIG.11B

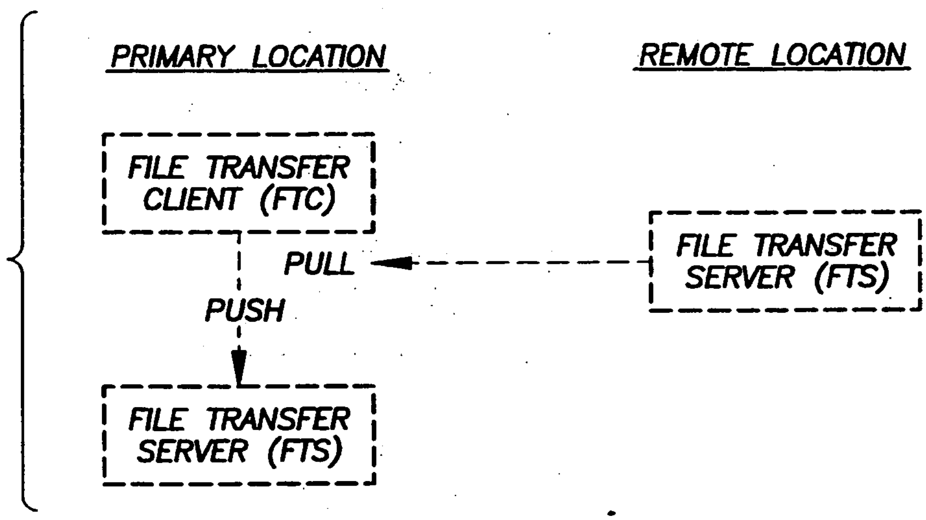
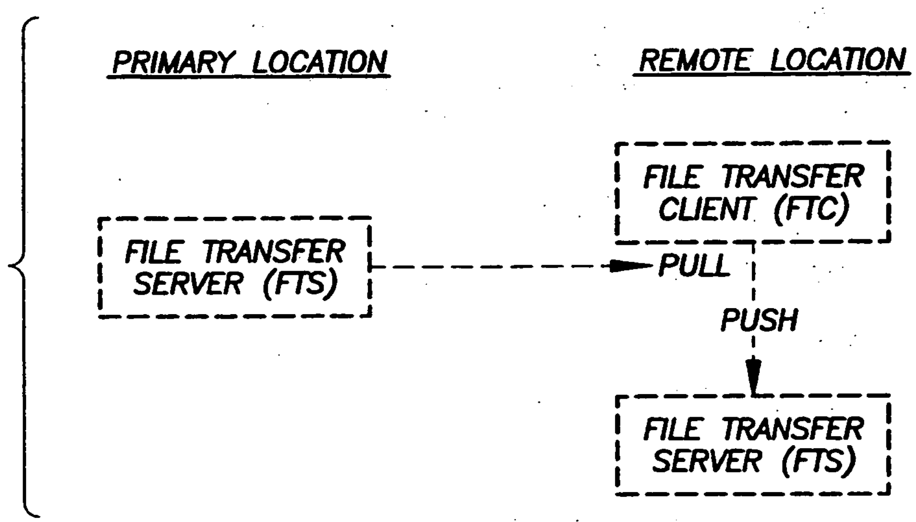


FIG.11C



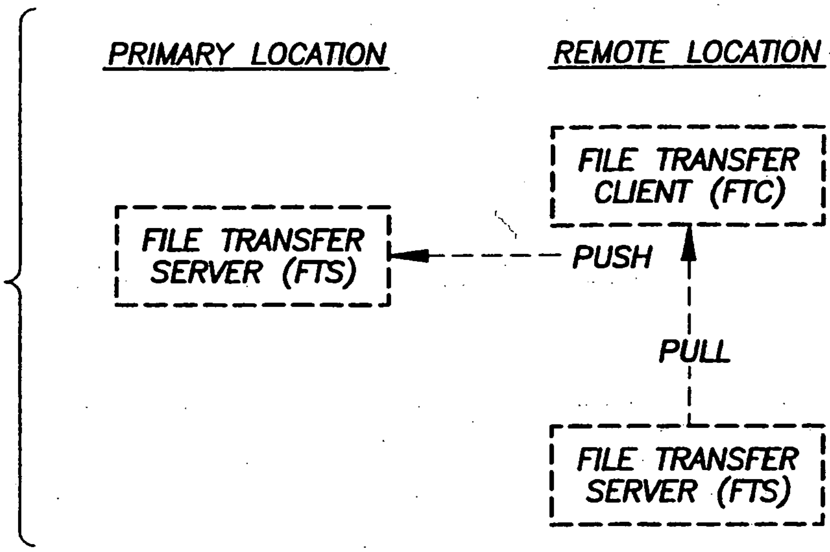


FIG. 11D

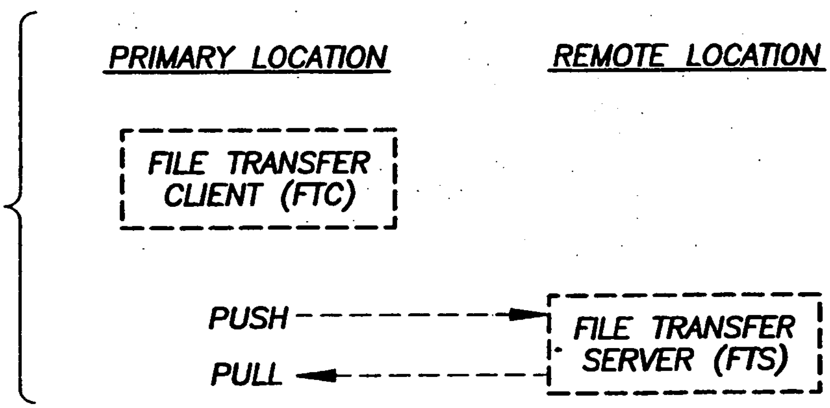


FIG. 11E

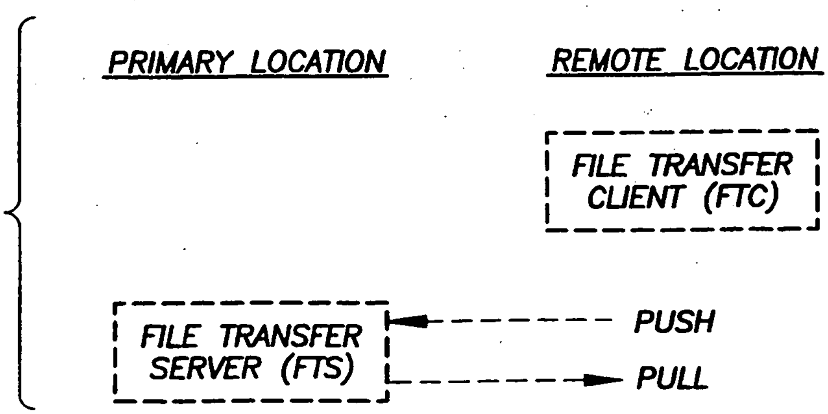


FIG. 11F

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 97/22688

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	POO G -S ET AL: "FILE RECOVERY BY ISO FTAM" COMPUTER COMMUNICATIONS, vol. 16, no. 12, 1 December 1993, pages 798-805, XP000411875	4,65,66
A	see page 798, column 1, line 1 - line 5 see page 798, column 3, line 5 - line 33 see page 799, column 3, line 20 - line 25	1-3,5-64
A	US 5 235 680 A (BIJNAGTE LEENDERT M) 10 August 1993 see abstract see column 1, line 11 - line 19 see column 3, line 18 - line 20 see column 8, line 55 - line 67 see column 10, line 3 - line 5	1-66

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 May 1998	05/06/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Adkhis, F

1

International application No
PCT/US 97/22688

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	US 4 686 625 A (BRYAN JOHN N) 11 August 1987 see abstract see column 1, line 7 - column 2, line 59 -----	1-66

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

Internation pplication No
PCT/US 97/22688

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5235680 A	10-08-1993	CA 1288516 A	03-09-1991
US 4686625 A	11-08-1987	NONE	